Universidade do Minho

Mestrado em Engenharia Informática

Projeto em Engenharia Informática

Servicify

Fábio Senra A82108
Jaime Leite A80757
João Freitas A74814
João Pimentel A80874
Nuno Rei A81918
Paulo Barbosa A82886
Paulo Bento A81139
Pedro Gonçalves A82313

19 de janeiro de 2021

Resumo

A prestação de serviços sempre foi uma variável contínua na quotidiano das pessoas. No entanto, ainda é algo que ocorre, maioritariamente, sem auxílio de uma ferramenta tecnológica para facilitar o processo. Assim, o objetivo deste projeto passou por desenvolver uma plataforma que se destacasse dos concorrentes pelas suas funcionalidades e estratégias de *marketing*, pois, apesar de estes existirem, são pouco conhecidos e de utilização pouco amigável. Consequentemente, foi importante garantir que a plataforma simplificasse a requisição de serviços, melhorando a qualidade de vida de todos os utilizadores. Além disso, em termos financeiros, o estudo permitiu prever lucros consideráveis ao final de 4 anos em mercado. Estes foram modelados através do foco em campanhas de *marketing* abrangentes e direcionadas ao público em geral, uma boa gestão da publicidade na plataforma e da contratação de elementos estritamente necessários.

Quanto à plataforma desenvolvida, esta foi dividida em duas grandes componentes, *backend* e *frontend*, permitindo modular e simplificar os processos de manutenção e/ou melhoria. O primeiro foi desenvolvido usando a *framework NodeJS*, juntamente com *Express*, e o segundo usando *VueJS*. Ademais, foi efetuada a publicação da plataforma num servidor do Departamento de Informática, através do uso de *Docker*, estando esta disponível ao público.

Desta forma, foi desenvolvido um produto que teve por base as opiniões de possíveis utilizadores, tornando-o atrativo, simples e com perspetivas de sucesso. No entanto, de modo a realmente colocar este no mercado, certas alterações seriam necessárias, por exemplo, o uso do protocolo *HTTPS*, notificações, entre outras.

Conteúdo

Lista de Figuras

1 Introdução

Ao longo deste capítulo será apresentado o contexto do projeto, a sua motivação e os objetivos. Além disso, será, também, apresentada a estrutura dos restantes capítulos deste relatório.

1.1 Contexto

A prestação de serviços sempre foi uma variável contínua na história da humanidade. No entanto, ainda é algo que ocorre, maioritariamente, na sua forma mais pura, ou seja, sem uma ferramenta tecnológica para facilitar a sua requisição. Assim, vivendo num mundo que se torna mais dependente de tecnologia a cada dia que passa, facilitar este processo de fornecimento e requisição de serviços torna-se uma necessidade óbvia. A título de exemplo, imagine que pretende limpar a chaminé de sua casa, porém, não tem as ferramentas, nem as capacidades para o fazer. Assim, sem conhecer algum destes prestadores, perguntar a pessoas da sua zona, ou pesquisar online, a tarefa torna-se bastante complexa. Note-se, ainda, que este processo nem sempre funciona. Dito isto, a possibilidade de pesquisar diretamente numa plataforma onde existem vários prestadores para um determinado serviço permite reduzir exponencialmente o tempo de pesquisa, bem como garantir que o cliente pagará um preço justo com base no mercado. Além disso, poderá escolher o melhor prestador, tendo por base as avaliações dos seus trabalhos anteriores e, até mesmo, a localização de ambos. Quanto à perspetiva dos prestadores, o seu leque de clientes aumenta consideravelmente, devido à maior exposição proveniente do uso da plataforma, deixando de estarem dependentes apenas de quem já os conhece e usa os seus serviços.

1.2 Motivação e Objetivos

Em termos de objetivos da implementação desta plataforma, a ideia passa por permitir que os clientes tenham acesso aos serviços que necessitam, de forma simples e rápida, com base em avaliações dos vários prestadores qualificados, localização, preço de serviço, entre muitos outros fatores. Quanto aos prestadores em si, pretende-se que estes possam expor os seus serviços a uma maior população, de modo a aumentar os seus lucros e adquirir mais clientes. Note-se que, de modo a permitir a melhor experiência quer aos prestadores, quer às pessoas a quem é prestado o serviço, a aplicação exigirá uma avaliação de ambas as partes. Desta forma, torna-se possível expor situações menos positivas quer por parte dos clientes, quer por parte dos prestadores. Além disso, permite que, em caso de o serviço ser efetuado de forma positiva, o *score* do cliente ou do prestador possa subir, melhorando as suas chances de encontrar mais serviços.

Em jeito de resumo, o grande objetivo deste projeto passa por superar o estado da arte nesta temática, ou seja, desenvolver uma plataforma que se destaque dos concorrentes pelas suas funcionalidades. É, também, crucial que a plataforma permita simplificar a requisição de serviços, melhorando a qualidade de vida de solicitadores e prestadores.

1.3 Estrutura do Relatório

Este relatório está dividido em cinco capítulos principais. O presente capítulo introduz a motivação, contexto e o principal objetivo do projeto. No Capítulo 2 é apresentada a proposta de valor, modelo de negócio e requisitos do projeto. O Capítulo 3 descreve os vários passos de implementação da plataforma, desde a sua modelação até ao seu *deployment*. Posteriormente, no Capítulo 4, são apresentados os métodos de gestão de recursos utilizados no decorrer do projeto. Por fim, o Capítulo 5 descreve as conclusões principais e trabalho futuro.

2 Análise e Especificação

Neste capítulo será descrito, em maior detalhe, o projeto, nomeadamente, a proposta de valor do mesmo, a análise estratégica para promover a plataforma e o modelo de negócio.

2.1 Descrição do Projeto

Atualmente, o aparecimento de cada vez mais plataformas para efetuar processos bastante comuns no quotidiano das pessoas tem se tornado uma constante. Desta forma, a plataforma *Servicify* trata-se de um produto para facilitar a requisição e prestação de serviços, reduzindo o tempo despendido na procura dos melhores prestadores com base na categoria do serviço a realizar.

Assim, a plataforma foca-se tanto em melhorar a experiência de quem requer os serviços, como de quem os presta. Quanto aos primeiros, é possível facilitar a procura de profissionais, de forma otimizada e simples, sem ser necessário estar dependente de conhecidos de terceiros, pesquisas *online*, entre outros aspetos. No que toca aos segundos, estes têm a possibilidade de aumentar, facilmente, a sua rede de clientes, bem como utilizar apenas um ambiente para se darem a conhecer ao mercado.

É, também, importante destacar a forma como a qualidade dos utilizadores é avaliada. Esta avaliação é feita através de um sistema de avaliações ponto a ponto, ou seja, o cliente avalia o prestador e, ao mesmo tempo, é avaliado pelo último. Tal permite que os serviços prestados com qualidade sejam premiados e mais expostos em pesquisas futuras, além de que permite verificar, em certa medida, a veracidade das avaliações. Ademais, permite que um prestador possa recusar um serviço com base no comportamento passado de um determinado cliente.

2.2 Proposta de Valor

Como mencionado anteriormente, os consumidores da plataforma serão tanto os prestadores de serviços, como os solicitadores dos mesmos. Deste modo, como se pode observar na Figura 1, o valor de uma plataforma deste tipo recai na poupança temporal e monetária. Isto permite, através da facilidade na procura e comparação de prestadores, garantias de qualidade com base em avaliações e alargamento de visibilidade dos prestadores.

Apesar da existência de algumas opções no mercado, plataformas desta natureza não são frequentemente utilizadas no quotidiano das pessoas. De modo a evitar este problema, contrariamente aos concorrentes, o projeto em questão permite uma utilização grátis e contacto entre clientes e prestadores, com possibilidade de negociação diretamente na plataforma. É, também, crucial destacar que a plataforma não terá limite de categorias de serviços, ou seja, uma categoria não existente até então pode ser adicionada, após verificação, permitindo um aumento no público alvo da plataforma. Além disso, contrariamente a todos os concorrentes, a *Servicify* possui um sistema de avaliações *end-to-end*, permitindo não só verificar a realização de um serviço, mas também criar um histórico comportamental dos solicitadores, caraterística essencial para que um prestador possa aceitar ou recusar a realização do serviço, com base nos seus ideais.

É, ainda, importante destacar a tendência de futuro de uma plataforma como esta, numa realidade cada vez mais direcionada à tecnologia.

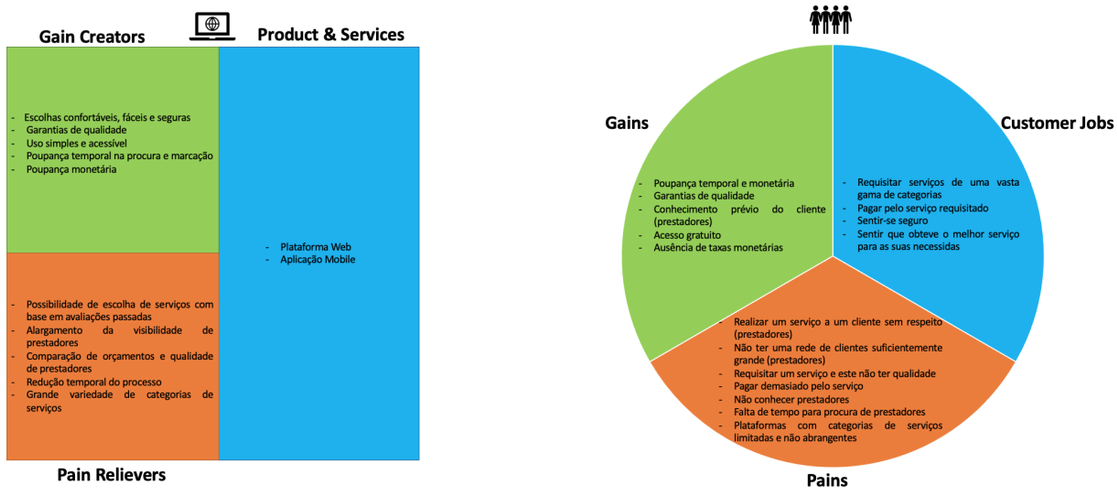

Figura 1 – Proposta de valor da plataforma.

2.3 Análise Estratégica

Como mencionado anteriormente, a grande missão da plataforma é permitir a requisição de serviços de forma simples, independentemente da categoria do serviço em questão. Além disso, é crucial garantir que os solicitadores possam escolher o prestador com base em vários parâmetros, como valor do serviço, qualidade do prestador (baseado em avaliações), localização, entre outros fatores que possam ser importantes.

Apesar de este tipo de plataformas não ser do conhecimento geral, existem concorrentes tal como referido anteriormente. Destes, podem ser destacados a plataforma *Olx* e *Craigslist*, ambos ambientes de partilha de anúncios, não direcionados a facilitar a realização de serviços. No entanto, plataformas como *Get Ninjas*, atuante do mercado brasileiro e mexicano, *Zaask* e *Fixando* foram construídas com a mesma base deste projeto. Todavia, nenhuma destas plataformas possui uma quantidade elevada de categorias de serviços, contrariamente ao *Servicify*, onde categorias podem ser adicionadas em qualquer momento, após validação. Além disso, é comum os prestadores terem que pagar para terem acesso e avaliarem os pedidos dos clientes, uma abordagem que pode facilmente baixar a taxa de aderência à plataforma. Ademais, as avaliações nestas plataformas apenas são efetuadas pelos clientes aos prestadores, perdendo-se a possibilidade de recusa de serviços por parte dos prestadores caso o cliente tenha demonstrado um perfil negativo anteriormente. De destacar, ainda, que estas avaliações não passam de texto e uma cotação entre 0 e 5, nem sempre ajudando a perceber, realmente, a qualidade dos envolvidos.

No que toca à estratégia da plataforma, a ideia passa por adquirir uma base de utilizadores considerável, permitindo aumentar a visibilidade da aplicação a curto prazo. Desta forma, os prestadores poderão, facilmente, aumentar a sua rede de negócio e os clientes terão acesso a cada vez mais opções. De modo a garantir uma maior adesão à plataforma, não podem ser implementadas táticas que possam afugentar os prestadores, como a necessidade de pagar para responder a clientes.

Assim, com base no *feedback* que pode ser obtido pela análise dos principais concorrentes, uma plataforma amigável aos seus utilizadores é a grande força deste projeto. A plataforma possui as bases para o sucesso uma vez que permite que os prestadores tenham conhecimento prévio do indivíduo a quem estão a realizar o serviço, e que possam orçamentar e negociar o valor do mesmo. Além destes aspetos, a existência de avaliações multi-campo e a facilitação da pesquisa de profissionais mais aptos para os interesses do cliente são fatores que poderão contribuir para o sucesso da mesma.

É, também, importante destacar que nenhuma das plataformas em comparação se estende a um território considerável, ou seja, todas elas se focam em um ou dois países, algo que o *Servicify* pretende globalizar, com o aumento da sua visibilidade e aderência. Apesar disso, a existência de plataformas semelhantes no mercado pode tornar-se um problema, tendo estas já uma base de utilizadores e não sendo universalmente utilizadas. Consequentemente, poderá haver alguma resistência de adesão quer de prestadores, quer de solicitadores.

Desta forma, a Figura 2 representa o *canvas* da análise estratégica da plataforma, retratando,

de forma sumariada, os aspetos mencionados ao longo desta secção.

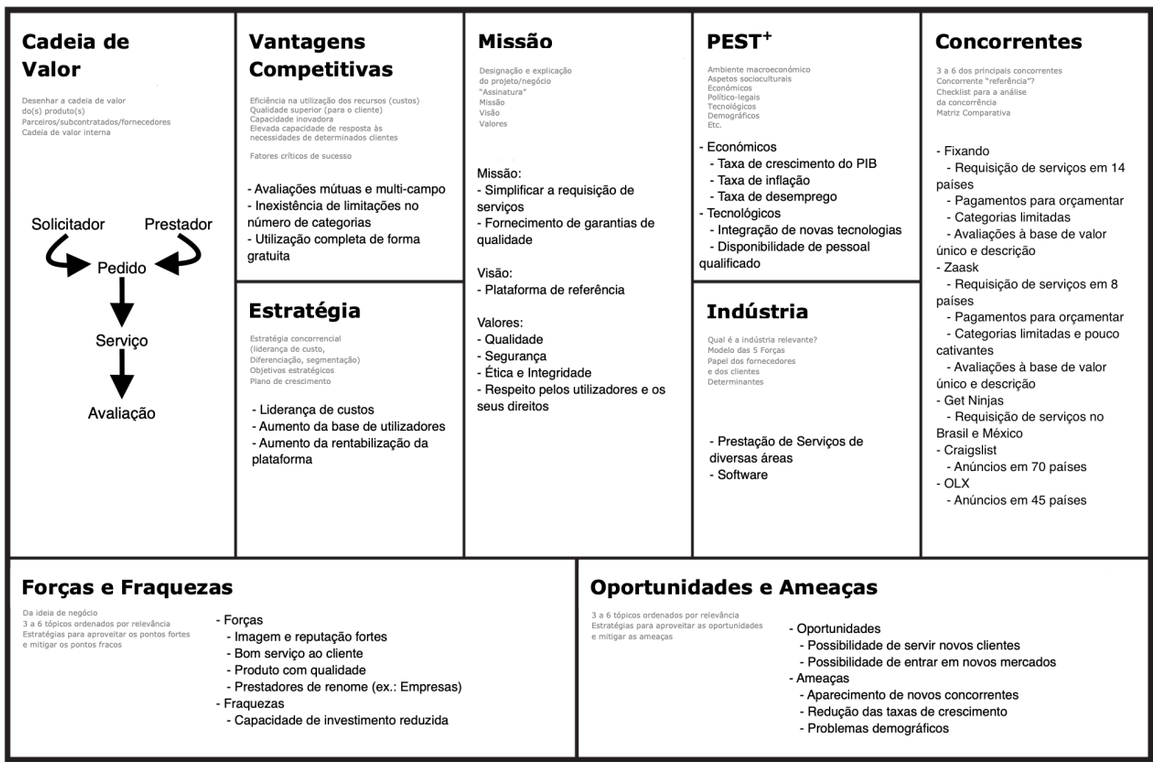

Figura 2 – Análise estratégica da plataforma.

2.4 Modelo de Negócio

A definição de um modelo de negócio permite que, numa fase inicial, sejam definidos elementos de sucesso, como o conceito do produto e o público alvo do mesmo.

Graças à utilização de um *canvas* para simplificar a leitura do modelo, como se pode ver pela Figura 3, alguns dos principais aspetos a retirar passam pelos problemas a resolver e quem beneficia dessa resolução. Além destes, é possível verificar como é criado valor para o cliente e como o produto chega a estes, como o negócio se manterá competitivo e a definição dos custos e receitas principais.

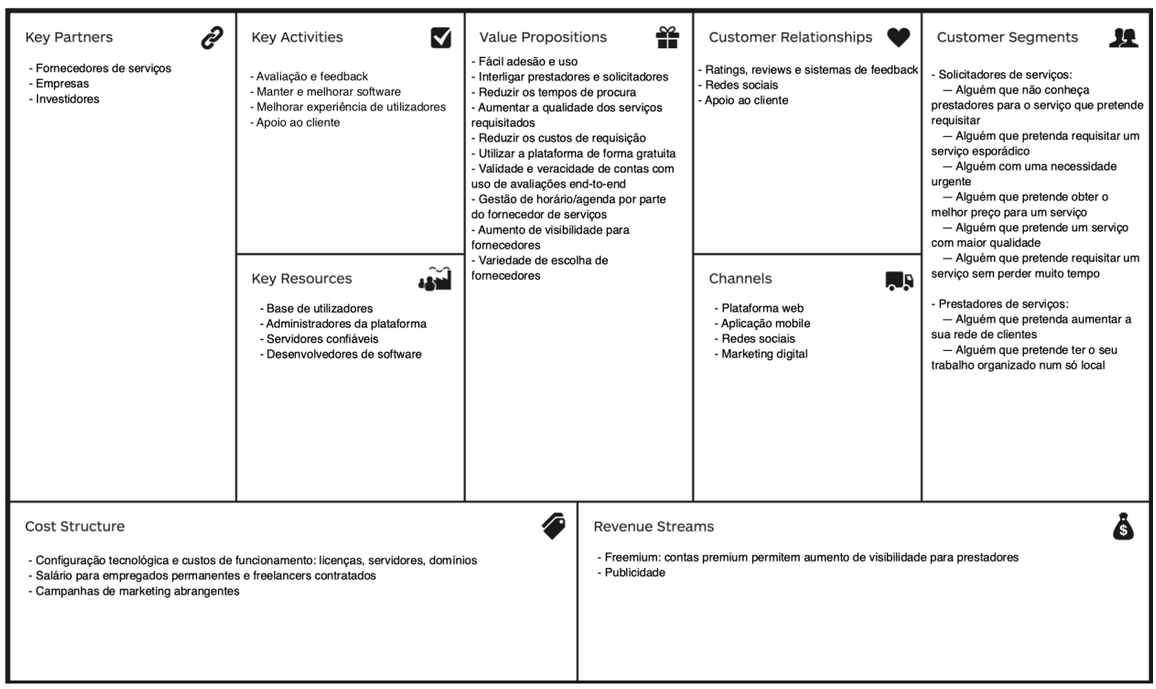

Figura 3 – Modelo de negócio da plataforma.

No que toca aos clientes alvo da plataforma, estes podem ser divididos em dois grupos. Por um lado, a plataforma será utilizada por pessoas que pretendem requisitar serviços, com foco especial em utilizadores que não conheçam prestadores para os serviços que pretendam ou tenham uma

necessidade urgente que possam publicitar. Além desses, utilizadores que tenham a intenção de obter o melhor preço possível beneficiam do uso da plataforma, uma vez que podem comparar preços e a qualidade de vários prestadores. É, ainda, bastante benéfica para pessoas que pretendam poupar tempo de procura, já que os resultados se encontram à distância de um botão, ao invés de um processo lento de marcações via telefone ou conversas com terceiros. Por outro lado, a aplicação é um porto de prestadores, sendo que estes aumentam a sua visibilidade e, consequentemente, a sua rede de clientes. Além disso, mantendo todos os seus serviços num só local, os prestadores acabam por ter a sua agenda e histórico organizados e com acesso simples e rápido.

Todas as vantagens mencionadas para estes utilizadores constituem a proposta de valor da plataforma, bem como a utilização completamente gratuita de todas as funcionalidades. Assim, a geração de receitas passa pela abordagem *freemium*, ou seja, a existência de contas *premium*, aumentando a visibilidade de prestadores e uso de publicidade na plataforma. Quanto aos canais, ou seja, a forma como a proposta de valor chega aos clientes, estes são constituídos por uma plataforma *web*, uma aplicação *mobile*, redes sociais para melhor comunicação com os clientes de modo a resolver problemas de forma simples e rápida, e *marketing* digital, permitindo que o público alvo tome conhecimento do produto. No que diz respeito às relações com os clientes, estes podem avaliar a plataforma com um sistema de cotações e *feedback*, bem como uso de redes sociais e apoio ao cliente.

Noutro tópico, os parceiros essenciais passam pelos próprios prestadores, já que uma plataforma desta natureza não sobrevive sem eles, empresas, podendo levar a que estas realizem uma grande parte dos seus serviços com auxílio da plataforma, e investidores, contribuindo para um possível desenvolvimento de mais funcionalidades e aumento de qualidade. Assim, ainda neste seguimento, os recursos chave são, além de administradores da plataforma, servidores confiáveis e desenvolvedores de *software*, ou seja, os utilizadores na sua globalidade. Desta forma, com o aumento da base de utilizadores, todos os elementos da proposta de valor aumentam de eficiência, sendo uma plataforma que ganha com o aumento de utilizadores. Por conseguinte, as atividades principais passam pela manutenção e melhoria do produto, tirando proveito de *feedback* fornecido pelos utilizadores, o que permite melhorar a experiência e fornecer apoio aos mesmos.

Por fim, todo este processo tem custos associados, desde uma configuração tecnológica, ou seja, contratação de máquinas para ter a plataforma em execução, até custos de funcionamento como licenças e domínios. Além disso, a resolução de *bugs* e desenvolvimento de novas funcionalidades implica a contratação de pessoal qualificado para tal, sendo necessário o pagamentos dos seus salários. É, ainda, importante destacar os custos associados a campanhas de *marketing* digital, de forma a dar a plataforma a conhecer às pessoas, contrariamente ao que acontece com os concorrentes, que não são de uma forma geral.

2.5 Análise Económico-Financeira

De modo a compreender a viabilidade económica da plataforma, uma análise económico financeira foi necessária, permitindo estimar as despesas e fontes de rendimento da plataforma nos primeiros 5 anos da sua existência no mercado. Esta análise teve como principal objetivo perceber quais os recursos necessários a alocar nas diferentes áreas da empresa e qual o impacto dessas alocações na saúde financeira do negócio. Desta forma, as receitas da plataforma são provenientes de duas fontes distintas. A primeira é referente à venda de subscrições *premium* aos clientes, sendo que, neste caso em concreto, foram assumidas três formas de pagamento: mensal (15), semestral (75) e anual (135). Todavia, a perspetiva de adesão no primeiro ano é realista, não esperando mais do que 15 subscrições mensais, 10 subscrições semestrais e 2 subscrições anuais. Quanto à segunda, esta é fruto da monetização da plataforma através do uso de publicidade, onde no primeiro ano seriam colocadas duas barras laterais com o valor fixo de 120 mensais e nos anos seguintes este uso já seria diferente, com base no número de cliques e outros aspetos. Consequentemente, a última trata-se da maior fonte de lucro para um negócio desta natureza, apesar de se tratar de uma fonte de rendimento mais volátil, quando em comparação com a venda de subscrições. Isto deve-se ao facto de ser dependente do número de visitas mensais que a plataforma recebe, ao invés de garantir a adesão de utilizadores. Quanto aos gastos, foram considerados valores como salários de colaboradores e

gastos de funcionamento da aplicação, até valores usados para promoção da plataforma, um dos principais pontos chaves no sucesso deste negócio, começando por ser um valor definido anualmente nos anos com prejuízo e passando a um valor percentual dos ganhos, a partir do momento em que a plataforma apresenta lucro.

É importante mencionar que esta análise foi efetuada com auxílio de um ficheiro *excel* fornecido pelos docentes da Unidade Curricular, no qual os diversos valores foram introduzidos, produzindo um modelo financeiro. Neste foi possível constatar que a empresa geraria lucro durante o quarto ano de funcionamento, um resultado expectável, considerando o elevado investimento inicial necessário para estabelecimento da marca no mercado. Este investimento inicial teria como principais fontes o capital dos sócios no valor de 35 000 e ainda financiamento bancário no valor de 290 000, de modo a garantir a subsistência da empresa nos 3 primeiros anos. Além disso, no terceiro ano seria necessário um novo investimento de 113 000 oriundos de financiamento bancário possibilitando assim o aparecimento de um lucro de 560 910 no quarto ano de funcionamento. Estes valores elevados podem ser justificados com a necessidade de uma forte política de *marketing* para promover a aplicação, visto este ser o *handicap* mais notório dos adversários diretos. É ainda de realçar que, durante os anos de funcionamento da empresa, são efetuados alguns investimentos em material para dar resposta à entrada de novos colaboradores e à manutenção dos dispositivos utilizados. Como em qualquer empresa foram também considerados os gastos com bens essenciais como água, luz, contabilidade subcontratada, assim como a renda do local onde esta se iria sediar. Por fim, o projeto apresenta um valor atual líquido de 2 776 081, uma taxa interna de retorno de 93.16% e, como referido anteriormente, um *payback* de 4 anos.

2.6 Levantamento de Requisitos

Uma das etapas essenciais no desenvolvimento de peças de *software* é a etapa de levantamento de requisitos, fornecendo informações valiosas sobre aspetos a implementar na plataforma a ser desenvolvida. Para tal, além de *brainstorming* e discussões entre elementos, a equipa disponibilizou um formulário com questões pertinentes para auxiliar o processo, podendo este ser analisado na secção 6.1. No que respeita aos requisitos em si, estes podem ser divididos em dois grandes grupos: requisitos funcionais e requisitos não-funcionais. Quanto aos últimos, estes podem, ainda, ser sub-divididos em diversas categorias.

Nesta sub-secção, começam por ser apresentados os requisitos funcionais em formato de cartão *Volere* simplificado, isto é, apresentando apenas os campos considerados mais relevantes. Desta forma, tornando cada cartão menos extenso, a leitura dos mesmos torna-se mais dinâmica. Em seguida, são, também, apresentados os requisitos não-funcionais.

No que toca à estrutura de cada cartão, esta é composta pelos seguintes campos:

- ID: identificador do requisito;
- Tipo: tipo do requisito consoante a divisão de *Volere*;
- Descrição: apresentação clara, concisa e objetiva do requisito;
- Justificação: esclarecimento do porquê do requisito existir;
- Critério de ajuste: quantificação do objetivo que se pretende alcançar com o requisito;
- Origem: forma/local de obtenção do requisito;
- Prioridade: importância da implementação deste requisito no projeto, seguindo o método de *MoSCoW* que divide os requisitos em 4 grupos:
    1. **Must:** Requisitos têm de ser obrigatoriamente implementados;
    2. **Should:** Requisitos que devem ser implementados;
    3. **Could:** Requisitos que poderiam ser implementados;
    4. **Won't:** Requisitos cuja implementação não está prevista dentro do espaço temporal do projeto.

Além disso, como principais *stakeholders* foram identificados os utilizadores que procuram alguém que lhes realize um dado serviço (solicitadores) e os utilizadores que prestam serviços (prestadores). Desta forma, o sistema é composto por, pelo menos, dois grupos de intervenientes alvo.

Em seguida apresentam-se os principais requisitos levantados, começando pelos que são comuns aos dois tipos de utilizadores alvo, seguidos pelos requisitos associados a cada entidade individual.

2.6.1 Requisitos comuns aos diferentes utilizadores

Os seguintes requisitos incidem essencialmente nas contas de cada utilizador, isto é, qualquer utilizador pode criar conta e alterar as caraterísticas do seu perfil.

Id: 1 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir criar conta.

Justificação: Ter acesso às funcionalidades da aplicação.

Critério de ajuste: O utilizador tem acesso a 100% das funcionalidades disponibilizadas.

Origem: Introspeção.

Prioridade: Must

Id: 2 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir editar o meu perfil.

Justificação: O utilizador pode querer alterar o perfil para, por exemplo, mudar a sua palavra-chave.

Origem: Introspeção.

Prioridade: Should

Para além de alterações na sua conta, existem funcionalidades que se revelam comuns, tanto para os solicitadores de serviços, como para quem os presta. Essas funcionalidades são: indicar o fim de serviços, cancelamento de serviços e, finalmente, a avaliação de serviços conforme indicam os requisitos abaixo indicados.

Id: 3 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir indicar o término do serviço.

Justificação: As duas partes do serviço indicam o fim para que o serviço seja validado como terminado.

Origem: Introspeção.

Prioridade: Must

Id: 4 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir cancelar um serviço.

Justificação: Indisponibilidade de um utilizador para realização do serviço.

Critério de ajuste: O cancelamento deve ser indicado, no mínimo, com 1 dia de antecedência.

Origem: Introspeção.

Prioridade: Must

Id: 5 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir avaliar o serviço.

Justificação: O utilizador pretende dar *feedback* do serviço realizado.

Origem: Questionário.

Prioridade: Must

Através do inquérito realizado, foi possível constatar que a possibilidade de visualizar faturas associadas aos vários serviços de cada utilizador se mostrou um aspeto importante. Desta forma, ao verem as informações de um serviço realizado, os utilizadores podem ver, caso exista, a fatura do mesmo.

Id: 6 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir visualizar as faturas a mim associadas.

Justificação: O utilizador pretende ver as faturas associadas ao seu nome.

Origem: Questionário.

Prioridade: Should

2.6.2 Requisitos dos solicitadores de serviços

Como se pode observar pelo requisito número 7, de modo a dar início ao processo de requisição de um serviço, é necessário que o solicitador de serviço comece por escolher a categoria onde pretender ver a sua necessidade suprimida.

Id: 7 **Tipo:** Funcional

Descrição: Como solicitador de serviço devo conseguir escolher a categoria do serviço.

Justificação: O utilizador pretende ver os prestadores de serviços apenas de uma determinada categoria.

Prioridade: Must

Finita a escolha da categoria, devem ser apresentados os prestadores associados ao solicitador os prestadores mais indicados para o trabalho.

Id: 8 **Tipo:** Funcional

Descrição: Como solicitador de serviço devo conseguir ver os prestadores de serviço.

Justificação: O utilizador pretende escolher um prestador de serviço.

Prioridade: Must

Continuando o processo, ou seja, quando apresentada a lista de prestadores de serviço, é necessário que o solicitador do mesmo analise a lista da forma que melhor lhe convém, para isto, é necessário fornecer mecanismos de filtragem da lista por diversos aspetos.

Id: 9 **Tipo:** Funcional

Descrição: Como solicitador de serviço devo conseguir filtrar os prestadores de serviço.

Justificação: O utilizador pode querer ver os prestadores de serviços ordenados, por exemplo, por melhor classificação.

Prioridade: Should

Depois da análise à lista de prestadores de serviços, o solicitador deve escolher o prestador que deseja e, em seguimento, enviar o pedido de serviço.

Id: 10 **Tipo:** Funcional

Descrição: Como solicitador de serviço devo conseguir enviar um pedido de serviço para um prestador.

Justificação: O utilizador pretende descrever o serviço que pretende ver realizado.

Prioridade: Must

Nem sempre o solicitador encontrará exatamente o prestador de serviço que pretende e, desta forma, poderá pretender fazer um pedido urgente de serviço inserido numa determinada categoria.

Id: 11 **Tipo:** Funcional

Descrição: Como solicitador de serviço devo conseguir criar um pedido urgente de serviço.

Justificação: O utilizador não consegue encontrar um prestador de serviço para a sua necessidade urgente.

Prioridade: Could

Numa fase posterior da construção do sistema e quando já existir uma base consistente de utilizadores, é visto como uma vantagem a introdução de pagamentos *online* para que este se torne mais rentável e amigável aos utilizadores. Como numa fase prematura da aplicação, a angariação de utilizadores é fulcral para que esta tenha uma boa visibilidade, a introdução de pagamentos nesta fase poderia repelir alguns utilizadores, como foi possível constatar pela realização de um questionário.

Id: 12 **Tipo:** Funcional

Descrição: Como solicitador de serviço devo conseguir pagar *online*.

Justificação: O utilizador pretende realizar o pagamento de forma mais cómoda.

Prioridade: Won't

2.6.3 Requisitos dos prestadores de serviços

No que toca às contas dos prestadores, estes podem, ainda, fazer um *upgrade* à sua conta para uma conta *premium*. Desta forma, o seu perfil passa a ser mais visível para os utilizadores da aplicação.

Id: 13 **Tipo:** Funcional

Descrição: Como prestador de serviço devo conseguir atualizar a minha conta para *premium*.

Justificação: O utilizador pretende receber benefícios *premium*.

Prioridade: Should

De forma a tornar a aplicação o mais abrangente possível, todas as categorias são bem-vindas e, por isso, os prestadores podem sugerir novas categorias que considerem que se encaixem melhor no seu serviço. De modo a garantir qualidade na plataforma, este pedido de inserção está sujeito a confirmação posterior de um administrador da aplicação.

Id: 14 **Tipo:** Funcional

Descrição: Como prestador de serviços devo conseguir realizar pedido para adição de nova categoria.

Justificação: O utilizador não encontra a categoria onde se encaixa.

Prioridade: Should

Para dar seguimento ao pedido de serviço enviado pelo solicitador é necessário que o prestador aceite ou recuse esse pedido, como se pode ver pelo requisito 15.

Id: 15 **Tipo:** Funcional

Descrição: Como prestador de serviço devo conseguir aceitar um pedido de serviço.

Justificação: O utilizador pretende realizar esse serviço.

Prioridade: Must

Id: 16 **Tipo:** Funcional

Descrição: Como prestador de serviço devo conseguir recusar um pedido de serviço.

Justificação: O utilizador não quer realizar esse serviço.

Prioridade: Must

Após a aceitação de um serviço, o prestador de serviço dá início ao processo de orçamentação, enviando um primeiro valor, tendo em conta a descrição anteriormente enviada no pedido de serviço.

Id: 17 **Tipo:** Funcional

Descrição: Como prestador de serviço devo conseguir indicar o orçamento do serviço.

Justificação: O utilizador pretende indicar o valor que prevê para o serviço.

Prioridade: Should

No seguimento da possibilidade de visualização de uma fatura de um serviço, o prestador deve ter a possibilidade de associar o documento referente à mesma ao serviço em questão. Desta forma,

a plataforma passa a ser uma ferramenta ainda mais útil para os seus utilizadores, possuindo documentos em formato digital, não sendo necessário manter registos em papel de faturas.

Id: 18 **Tipo:** Funcional

Descrição: Como prestador de serviço devo poder associar uma fatura a um serviço realizado.

Justificação: O utilizador pretende facilitar o processo de fornecimento de faturas ao solicitador.

Origem: Questionário

Prioridade: Should

Além disso, já com todos os pormenores acertados sobre os serviços a serem realizados, o prestador de serviço deve poder aceder a todos os serviços por realizar e realizados.

Id: 19 **Tipo:** Funcional

Descrição: Como prestador de serviço devo conseguir visualizar serviços realizados e a realizar.

Justificação: O utilizador pretende gerir as suas marcações.

Origem: *Brainstorming*

Prioridade: Should

2.6.4 Gestão do processo de negociação do orçamento

No que toca ao processo de orçamento, após o prestador de serviços indicar o valor do orçamento podem ocorrer vários cenários:

- Solicitador de serviço aceita imediatamente o orçamento.
- Solicitador de serviço decide negociar o orçamento, ficando sujeito a posterior aceitação/negociação/recusa por parte do prestador de serviço.
- Solicitador de serviço recusa imediatamente o orçamento

Desta forma é possível reparar que tanto o prestador de serviço, como quem o solicita, pode aceitar, recusar ou negociar o orçamento. Por consequência os seguintes requisitos são comuns aos dois tipos de intervenientes.

Id: 20 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir aceitar orçamento do serviço.

Prioridade: Must

Id: 21 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir recusar orçamento do serviço.

Prioridade: Must

Id: 22 **Tipo:** Funcional

Descrição: Como utilizador geral devo conseguir negociar orçamento do serviço.

Justificação: O utilizador pretende ajustar o valor do serviço.

Prioridade: Could

De realçar que neste processo está ainda implícita a negociação da data, hora e tempo estimado para realização do serviço.

2.6.5 Requisitos de administração

Como já foi supracitado, existe a possibilidade de adição de novas categorias, mas nem todos os pedidos poderão ser atendidos. Assim, existe a necessidade de garantir que nem todos esses pedidos geram novas categorias e desta forma passam pela decisão de um administrador do sistema.

Id: 23 **Tipo:** Funcional

Descrição: Como administrador do sistema devo conseguir resolver os pedidos de adição de categorias.

Justificação: Um administrador pretende melhorar o sistema para que chegue a mais utilizadores.

Origem: *Brainstorming*

Prioridade: Should

Outro aspeto importante para a administração de qualquer sistema é a possibilidade de visualização das métricas, de forma a ter uma visão mais detalhada dos vários componentes do sistema.

Id: 24 **Tipo:** Funcional

Descrição: Como administrador do sistema devo conseguir aceder às métricas da plataforma.

Justificação: Um administrador pretende saber quais as funcionalidades que são mais acedidas.

Origem: *Brainstorming*

Prioridade: Should

Por fim, por forma a garantir que a plataforma contém um ambiente seguro e amigável a qualquer pessoa, bem como sempre atualizado, os administradores devem ter acesso a todas as funcionalidades do sistema. Assim, podem apagar, desativar e alterar níveis de acesso a utilizadores.

Id: 25 **Tipo:** Funcional

Descrição: Como administrador do sistema devo conseguir apagar, desativar e/ou alterar os níveis de acesso de utilizadores.

Justificação: O administrador pretende gerir os utilizadores da aplicação para manter a segurança.

Origem: *Brainstorming*

Prioridade: Should

2.6.6 Aspeto e sentimentos

No desenvolvimento de aplicações, pensar de antemão no aspeto do sistema permite poupar muito tempo, bem como garantir um certo nível de associação da palete de cores à própria marca.

Id: 26 **Tipo:** Não funcional

Descrição: Toda a aplicação deve estar em tons de azul e branco.

Justificação: As cores devem estar em concordância com o logótipo.

Origem: *Brainstorming*

Prioridade: Must

No seguimento da definição de cores, também a presença do logótipo deve estar sempre presente. Desta forma, os utilizadores têm sempre em mente a marca associada à ferramenta em uso.

Id: 27 **Tipo:** Não funcional

Descrição: O logótipo deve estar sempre presente na aplicação.

Justificação: Dar visibilidade à marca.

Critério de ajuste: O logótipo deverá aparecer em todas as páginas da aplicação.

Origem: *Brainstorming*

Prioridade: Must

Com base na grande variedade de plataformas atualmente, ter garantias de qualidade e fiabilidade torna-se fulcral. Assim, a grande maioria dos utilizadores da aplicação devem considerar a mesma como segura.

Id: 28 **Tipo:** Não funcional

Descrição: A aplicação deve parecer fiável para os seus utilizadores.

Critério de ajuste: Depois de ter contacto com a aplicação, 70% dos seus utilizadores devem concordar que a aplicação é fiável.

Origem: *Brainstorming*

Prioridade: Must

2.6.7 Usabilidade

De forma a garantir uma aderência elevada ao sistema, há uma necessidade de as funcionalidades da plataforma serem intuitivas para os seus utilizadores.

Id: 29 **Tipo:** Não funcional

Descrição: A aplicação deve ser intuitiva para um utilizador com idade entre os 20 anos e os 50 anos.

Critério de ajuste: Um utilizador deve conseguir utilizar 90% das funcionalidades após 20 minutos de utilização.

Origem: Introspeção

Prioridade: Should

Eventualmente, aquando da expansão da plataforma a várias regiões, torna-se necessário ter uma grande diversidade de línguas. No entanto, numa fase inicial é mais importante criar uma boa base de utilizadores, podendo a plataforma estar só em funcionamento numa língua que englobe um elevado número de possíveis usuários.

Id: 30 **Tipo:** Não funcional

Descrição: A aplicação deve permitir a escolha do idioma.

Justificação: A aplicação deve estar disponível em Português, Inglês, Espanhol, Francês, Alemão.

Origem: *Brainstorming*

Prioridade: Won't

Ainda no seguimento de garantir uma utilização simples, qualquer tarefa deve ser realizada em menos de 5 minutos. Além disso, a aplicação deve estar escrita de forma simples, permitindo que qualquer utilizador com um nível de escolaridade relativamente baixo consiga compreender todos os processos.

Id: 31 **Tipo:** Não funcional

Descrição: A aplicação deve necessitar de poucos passos para utilizar a maioria das funcionalidades.

Critério de ajuste: Depois de utilizar a aplicação durante duas semanas, qualquer tarefa deve ser realizada em 4/5 minutos.

Origem: Introspeção

Prioridade: Should

Id: 32 **Tipo:** Não funcional

Descrição: A aplicação deve estar escrita numa linguagem comum para os seus utilizadores.

Critério de ajuste: Qualquer pessoa com 6º ano de escolaridade ou superior deve perceber qualquer palavra ou ícone presente na aplicação.

Origem: Introspeção

Prioridade: Must

2.6.8 Performance

Tendo em conta a elevada dependência associada a este tipo de plataformas, é necessário que os pedidos efetuados à mesma sejam respondidos de forma rápida e durante a grande maioria do ano.

Id: 33 **Tipo:** Não funcional

Descrição: A aplicação deve ser rápida.

Critério de ajuste: Nenhum pedido à aplicação deve exceder os 5 segundos.

Origem: *Brainstorming*

Prioridade: Should

Id: 34 **Tipo:** Não funcional

Descrição: A aplicação deve estar disponível 361 dias no ano.

Critério de ajuste: A aplicação deve estar disponível 99% do tempo.

Origem: *Brainstorming*

Prioridade: Must

2.6.9 Operacional

Tratando-se de uma aplicação *web*, garantir que esta funciona nos principais *browsers* é extremamente importante, uma vez que qualquer dispositivo tem acesso a, pelo menos, um destes motores de busca. Além disso, a plataforma deve manter-se em funcionamento após cada atualização.

Id: 35 **Tipo:** Não funcional

Descrição: A aplicação deve funcionar nos 4 principais *browsers*.

Critério de ajuste: A aplicação deve funcionar em *Google Chrome*, *Firefox*, *Safari* e *Microsoft Edge*.

Origem: Introspeção

Prioridade: Could

Id: 36 **Tipo:** Não funcional

Descrição: A aplicação deve continuar a funcionar após cada atualização.

Justificação: As funcionalidades anteriores à atualização devem continuar a funcionar.

Origem: *Brainstorming*

Prioridade: Must

Numa fase posterior, permitir o acesso à plataforma deverá ser possível através de uma aplicação *mobile*. Assim, para os utilizadores que acedam com regularidade ao sistema nos seus telemóveis, deixa de ser necessário utilizar um motor de busca. No entanto, como mencionado anteriormente, todos os sistemas têm acesso a *browsers*, pelo que esta funcionalidade não se torna fulcral nesta fase.

Id: 37 **Tipo:** Não funcional

Descrição: A aplicação deverá poder ser acedida via aplicação *mobile IOS* ou *Android*.

Justificação: Tornar o acesso à aplicação mais confortável.

Origem: *Brainstorming*

Prioridade: Won't

2.6.10 Manutenção e suporte

Mais uma vez, tendo em conta a natureza do sistema, é crucial que este funcione como esperado, desde que o dispositivo tenha acesso à *internet* e a um motor de busca.

Id: 38 **Tipo:** Não funcional

Descrição: A aplicação deve funcionar em qualquer sistema operativo.

Justificação: A aplicação deve funcionar desde que haja um *browser* com acesso à *internet*.

Origem: *Brainstorming*

Prioridade: Must

2.6.11 Segurança

Assumindo a existência de legislação para proteção de dados individuais, os utilizadores devem ter a possibilidade de permitir ou recusar a reveleção de algumas das suas informações a outros membros da plataforma.

Id: 39 **Tipo:** Não funcional

Descrição: A aplicação deve dar ao utilizador a possibilidade de revelar ou não os seus dados como nome, morada, número de telemóvel.

Justificação: O utilizador decide se quer ou não revelar os seus dados.

Origem: *Brainstorming*

Prioridade: Must

Aquando da adição de pagamentos na plataforma, a informação bancária de cada utilizador deve estar devidamente protegida. Porém, não se tratando de uma das funcionalidades fulcrais nesta fase, não é um dos requisitos a implementar.

Id: 40 **Tipo:** Não funcional

Descrição: A aplicação mantém as contas bancárias dos utilizadores secretas.

Justificação: Apenas o utilizador tem acesso às informações da sua conta bancária.

Origem: *Brainstorming*

Prioridade: Won't

2.6.12 Cultural

Em termos culturais, não deve existir qualquer tipo de conteúdo ofensivo ou discriminatório com base em religiões ou etnias.

Id: 41 **Tipo:** Não funcional

Descrição: A aplicação não deve ser ofensiva para religiões ou etnias.

Origem: Introspeção

Prioridade: Must

2.6.13 Legalidade

Seguindo as boas práticas de qualquer plataforma com utilizadores, estes devem ter acesso às políticas de funcionamento e possíveis alterações nas mesmas.

Id: 42 **Tipo:** Não funcional

Descrição: A aplicação deve informar os seus utilizadores sobre a sua política de funcionamento.

Justificação: Todos os utilizadores devem conhecer os termos de funcionamento da aplicação.

Origem: Introspeção

Prioridade: Should

Id: 43 **Tipo:** Não funcional

Descrição: A aplicação deve informar sobre alterações na política de funcionamento.

Justificação: Todos os utilizadores devem conhecer a nova política de funcionamento da aplicação.

Origem: Introspeção

Prioridade: Should

3 Concepção da Resolução

No presente capítulo será descrita, em maior detalhe, a solução concebida para o projeto, ou seja, será explicado como o projeto vai ficar estruturado e, recorrendo a alguns diagramas, como será o funcionamento do mesmo.

3.1 Modelação do Sistema

Ao longo deste capítulo são apresentados detalhadamente os vários passos da modelação efetuada para simplificar o processo de implementação. Desta forma, foram modelados os diagramas de *Use Case*, arquitetura do sistema, máquinas de estado principais, entre outros.

3.1.1 Diagrama de *Use Cases*

Após terminado o processo de recolha dos requisitos funcionais, estando estes descritos no capítulo 2.6, torna-se possível elaborar um diagrama de *Use Cases,* de forma a representar as principais funcionalidades da aplicação de forma mais esclarecedora. Assim, observando a Figura 4, podem-se visualizar as diversas funções possíveis de efetuar por cada tipo de utilizador. A título de exemplo, os administradores de sistema podem adicionar novas categorias de serviços, visualizar métricas da plataforma, entre outras. Também é importante destacar que existem funcionalidades associadas a requerentes e prestadores, podendo estas ser vistas na categoria de *utilizador geral*.

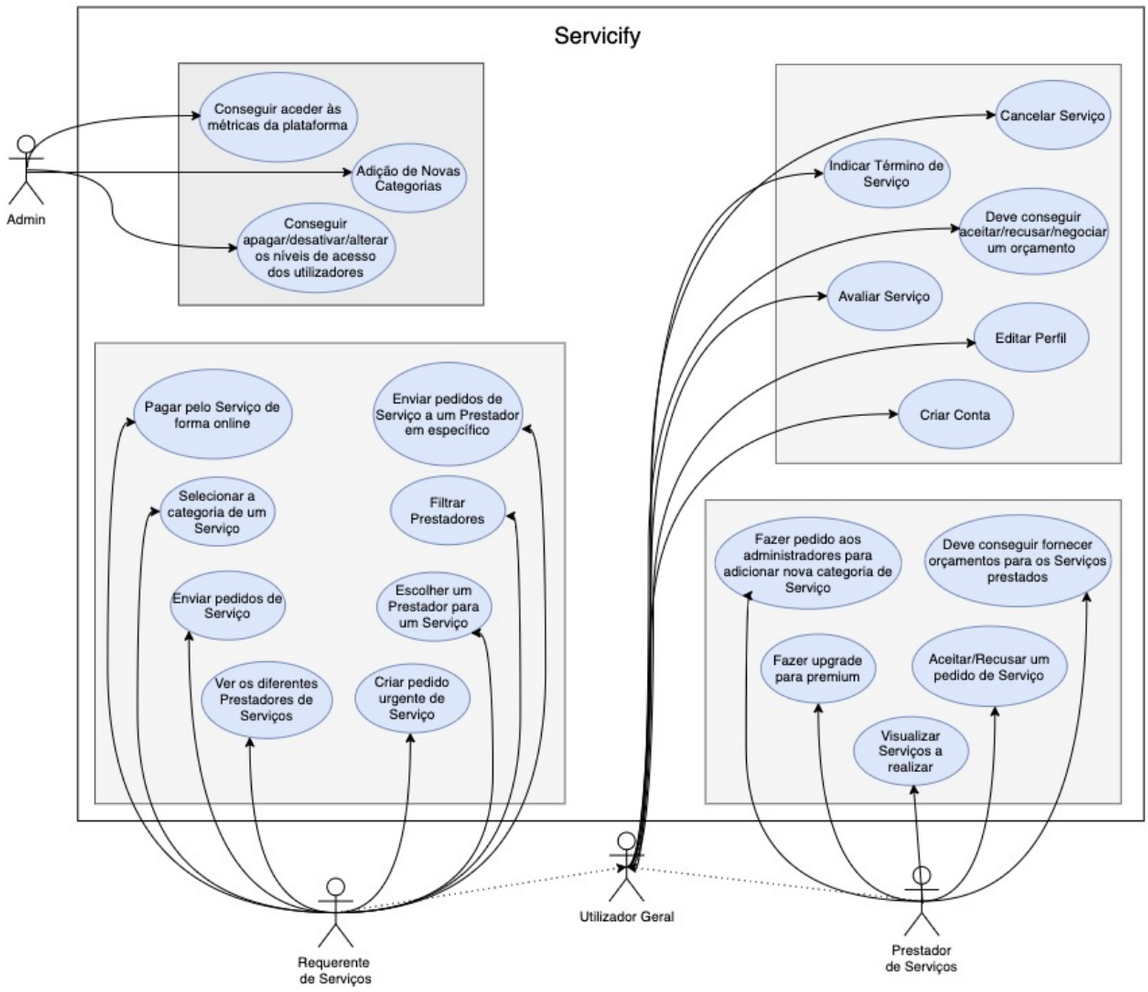

Figura 4 – Diagrama de *Use Cases* da plataforma.

3.1.2 Diagrama de Estados

Em seguida são apresentados alguns dos diagramas de estado mais importantes do sistema.

Visto que sistema é composto por dois grandes tipos utilizadores, prestadores e solicitadores de serviços, são necessários diagramas de estado para cada um destes tipos, como pode ser observado na Figura 5, onde está representada uma visão bastante geral da plataforma.

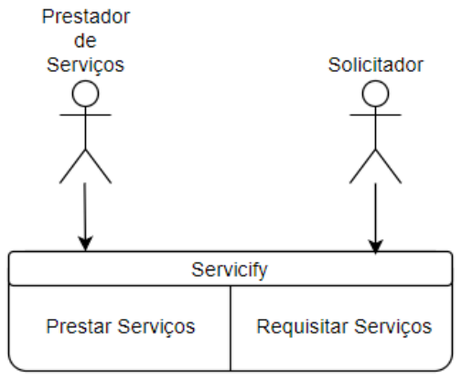

Figura 5 – Visão geral do sistema.

Começando pelos prestadores, estes apenas podem utilizar o serviço caso tenham uma conta já criada. Assim, após a criação da mesma, funcionalidades como editar o seu perfil, visualizar os seus serviços por realizar (agenda) e aceitar propostas de serviço feitas por solicitadores ficam disponíveis, como se vê na Figura 6.

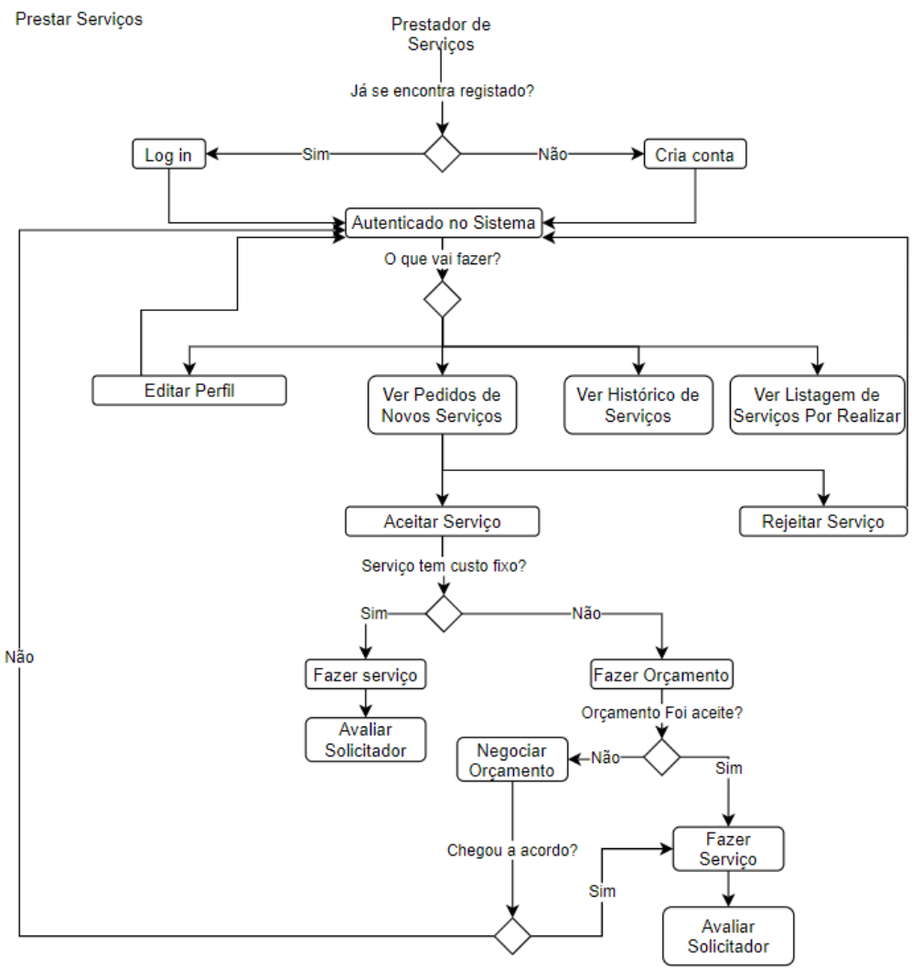

Figura 6 – Diagrama de estados referente aos prestadores de serviços.

Quanto à edição de perfil, esta é, maioritariamente, relacionada com a edição das categorias de serviços, uma vez que é uma caraterística exclusiva dos prestadores. Note-se que, caso alguma categoria não se encontre no sistema, o utilizador pode fazer um pedido de adição de categoria, devendo ser validado ou recusado pela administração, como se pode observar pela Figura 7. É, ainda, importante destacar o fluxo referente à negociação de um orçamento, onde, após aceitar um serviço, o prestador entra num estado de negociação do valor. Nesta situação, o prestador indica o seu valor inicial, podendo ser negociado pelo solicitador até que um dos intervenientes aceite ou recuse completamente o serviço. Finita esta etapa, efetua-se o serviço, ficando a faltar a avaliação mútua das partes, como forma de verificação e avaliação.

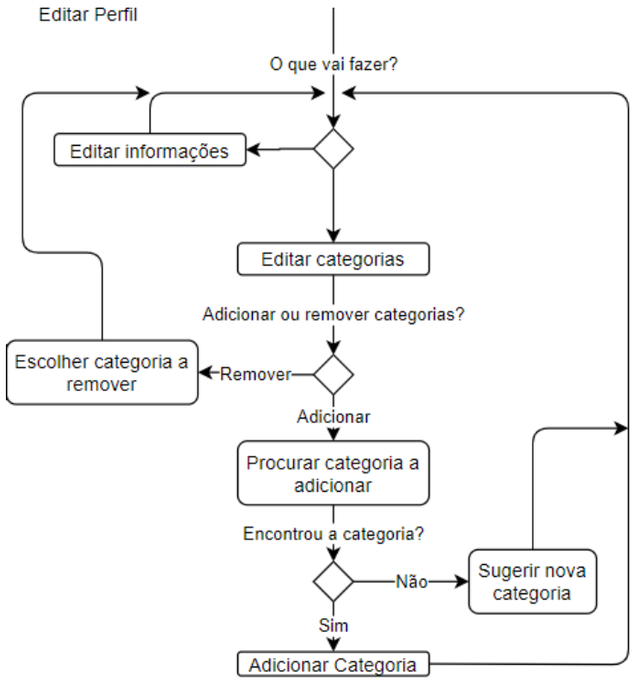

Figura 7 – Diagrama de estados referente à edição de perfil de prestadores.

No que toca aos solicitadores, estes podem consultar o sistema mesmo não estando autenticados. No entanto, para interagirem com os diversos utilizadores, ou seja, de modo a requisitarem serviços devem possuir uma conta. Desta forma, como se vê na Figura 8, após autenticados no sistema, os solicitadores podem procurar serviços com base em categorias e outros filtros pertinentes. Encontrando um prestador que esteja de acordo com as suas necessidades, podem ser encontrados dois cenários: existir um custo fixo ou ser necessário um pedido de orçamento. No primeiro caso, apenas é enviado um pedido de serviço, ficando dependente de aceitação do prestador. No segundo caso, é inicializado um processo de negociação até que se encontre um valor de acordo ou cancelamento do pedido. Finalmente, tal como no caso dos prestadores, após a realização do serviço, é efetuada uma avaliação mútua.

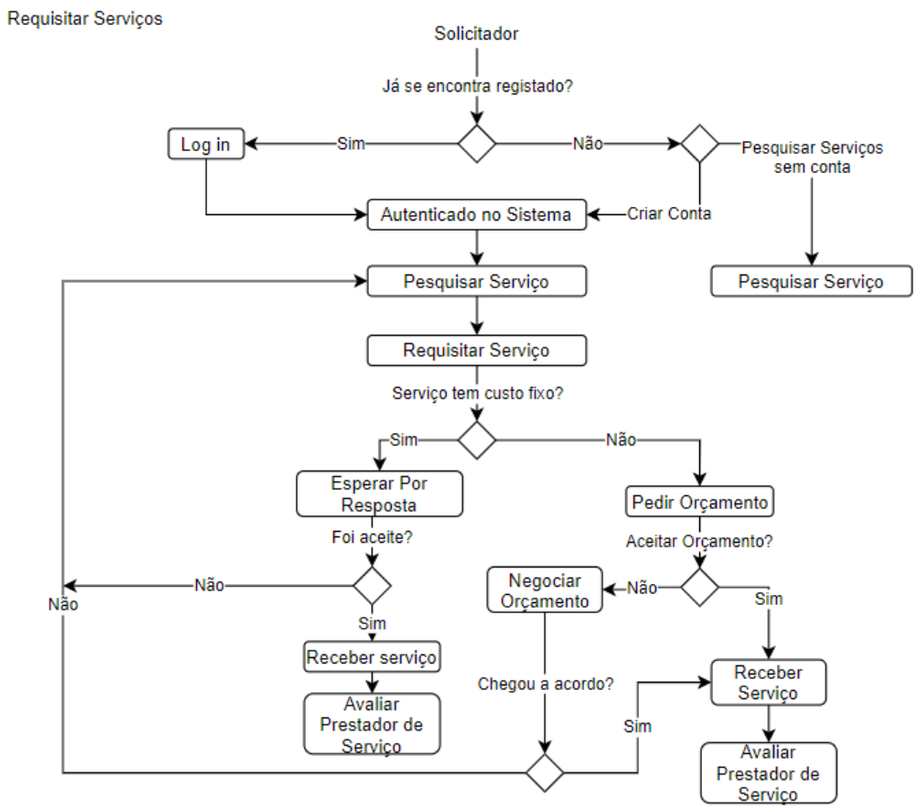

Figura 8 – Diagrama de estados referente aos solicitadores de serviços.

Além disso, é importante destacar que, tal como no caso dos prestadores, os solicitadores podem editar os seus perfis, nomeadamente as suas informações pessoais. Porém, de modo a manter o diagrama simples e focado nas funcionalidades que destacam a aplicação de uma plataforma genérica, este tipo de estados foi omitido.

3.1.3 Diagrama de Arquitetura do Sistema

De modo a compreender as interações dos utilizadores com o sistema, foi desenhado um diagrama de arquitetura de sistema, como se vê na Figura 9. Neste é possível observar que um utilizador pode aceder à plataforma via interface ou diretamente via *API* de dados. No caso do acesso ser feito pela interface, este funciona de forma bastante simples, onde a lógica e variáveis como *tokens* são controlados por uma aplicação escrita em *VueJS*. Desta forma, existe uma interação entre os dois servidores do sistema, sendo que o utilizador apenas lida com a interface. Porém, tendo em conta a existência de uma *API REST*, é possível fazer pedidos diretos a esta. Para tal, o utilizador necessita de utilizar o seu *token* ou chave de *API* nos pedidos, de forma a estar autenticado no sistema. Além disso, é importante destacar que, devido à sua facilidade e grande adesão por parte da comunidade de *developers*, o *backend* foi desenvolvido em *NodeJS* juntamente com *Express*. Por fim, graças à sua *performance* no que toca a conjuntos de dados de dimensão elevada, todos os dados são armazenados numa base de dados *NoSQL*, mais concretamente, *MongoDB*.

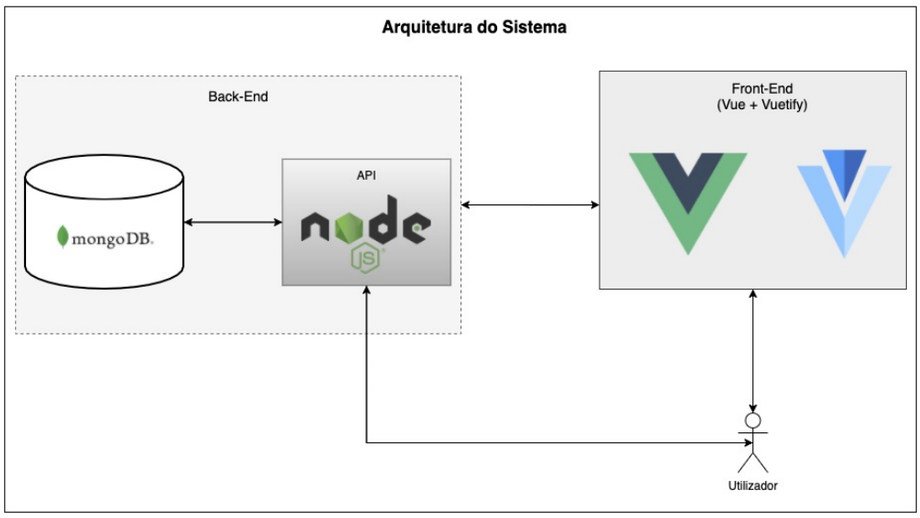

Figura 9 – Diagrama de arquitetura de sistema da plataforma.

3.1.4 Diagrama de Autenticação de Sistema

De modo a garantir uma implementação a ter em conta os vários métodos de autenticação no sistema, foi elaborado um diagrama referente a este processo, como se pode observar pela Figura 10. Nesta pode-se observar que as várias rotas da *API* de dados têm diferentes níveis de acesso a si associadas, pelo que estes devem ser sempre verificados. Além disso, as rotas podem ser acedidas quer com chaves, quer com *tokens* de utilizadores, tendo ambos um certo nível de acesso associado. É, ainda, importante destacar que no caso de os dados não conseguirem ser acedidos com base no nível possuído, um estado de erro é enviado ao utilizador. Ademais, tanto os *tokens*, como as chaves têm um tempo de vida a si associados, pelo que necessitam de estar válidos para serem utilizados.

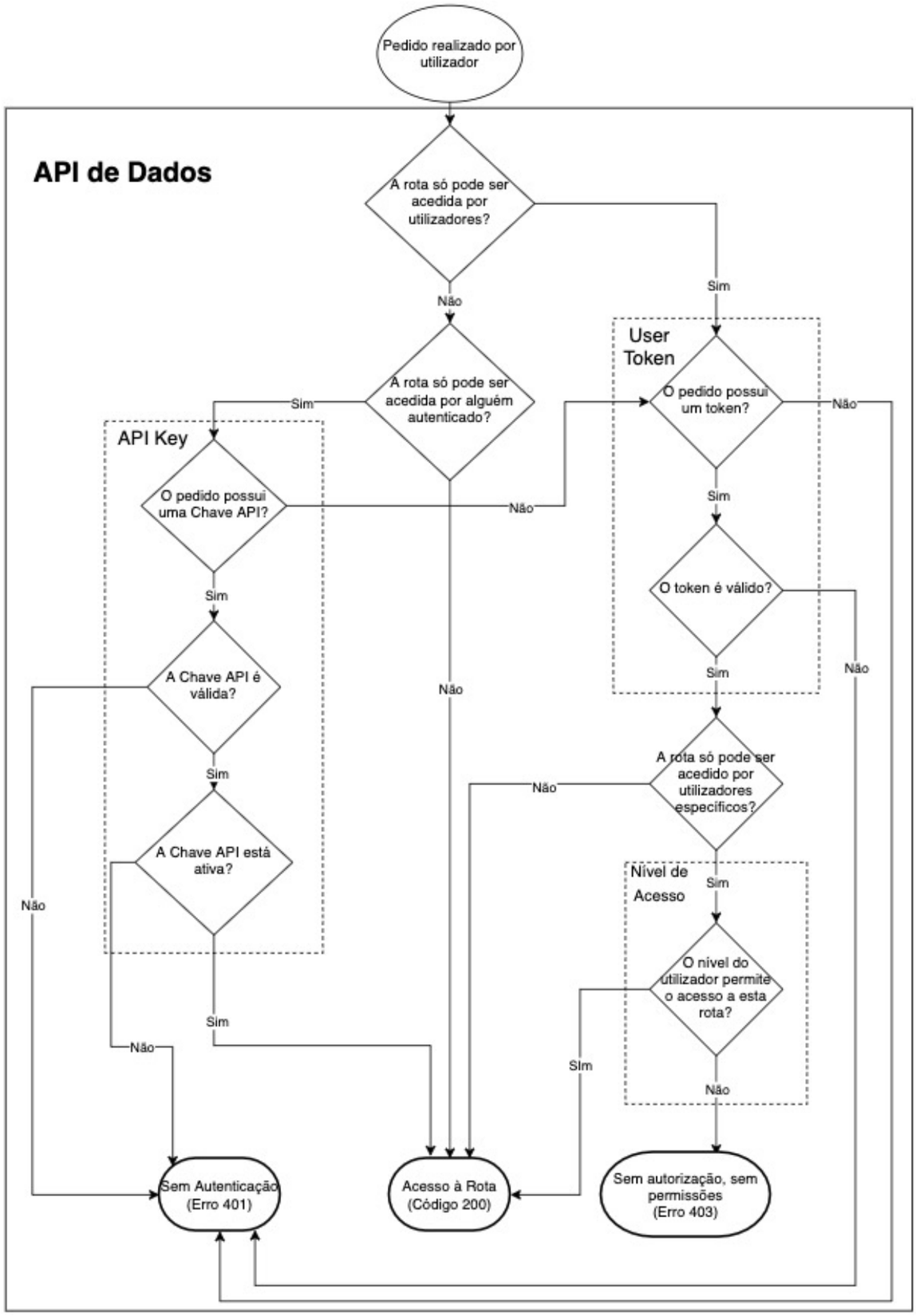

Figura 10 – Diagrama de autenticação no sistema.

3.1.5 *Mockups* da Aplicação

De modo a permitir uma implementação coesa da interface da plataforma, foram desenhados alguns *mockups*, retratando algumas das páginas principais. Assim, tendo por base os requisitos previamente levantados, a página inicial foi desenhada, tendo as cores e logótipo da aplicação presentes e em destaque, como se pode ver pela Figura 11.

Figura 11 – *Mockup* da página inicial da plataforma.

Seguindo a mesma ideia, na página de autenticação no sistema, visível na Figura 12, os utilizadores devem indicar o seu *email* e a sua palavra-passe.

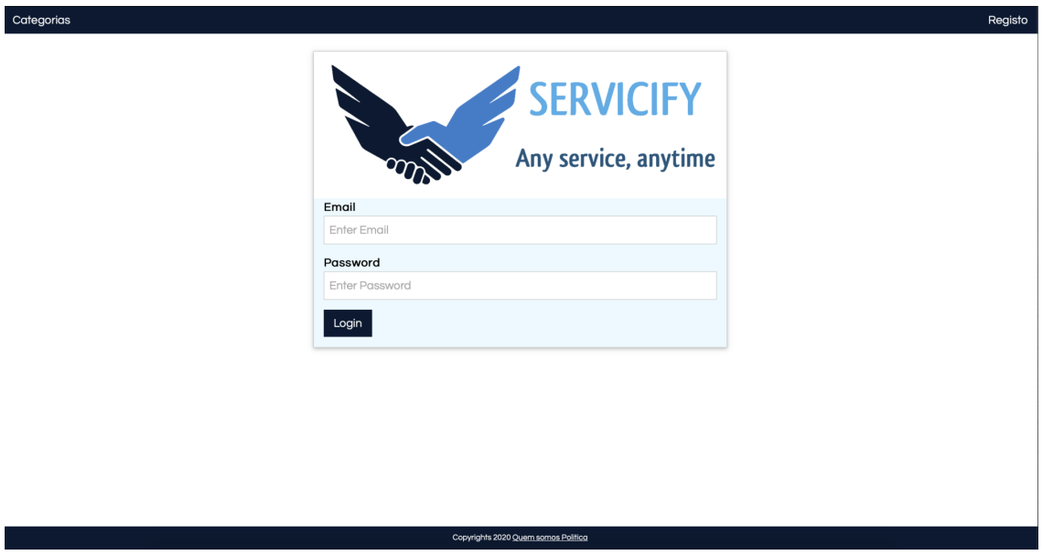

Figura 12 – *Mockup* da página de autenticação na plataforma.

Outro aspeto importante a modelar é a página de perfil de utilizadores, tendo esta a listagem dos vários serviços associados ao mesmo, como se vê na Figura 13.

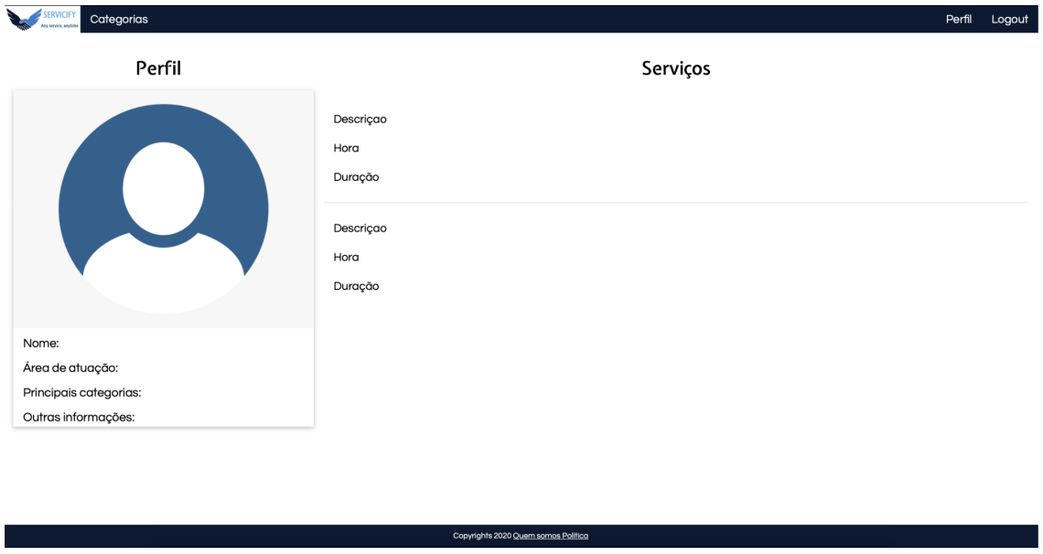

Figura 13 – *Mockup* da página de perfil na plataforma.

Por fim, é importante a existência de uma página para procurar categorias, como se pode observar pela Figura 14, mantendo-se um estilo constante por toda a aplicação.

Figura 14 – *Mockup* da página de listagem de categorias na plataforma.

3.2 Implementação do Sistema

Ao longo desta secção serão apresentados os métodos associados ao desenvolvimento de cada tarefa do projeto, bem como a forma de funcionamento destes, problemas encontrados e forma de resolver os mesmos.

3.2.1 Modelos de Dados

O primeiro componente a ser definido foi o modelo de dados geral da plataforma, permitindo a implementação de tarefas *CRUD* no *backend* da plataforma, ou seja, ações de criação (*create*), consulta (*read*), atualização (*update*) e remoção (*delete*).

Tendo por base os requisitos apresentados, o modelo de dados da plataforma é constituído por 6 grandes elementos, como se pode ver pela Figura 15, sendo estes, países, localizações, categorias, especializações, utilizadores e serviços.

Quanto aos dois primeiros, estes estão interligados, já que uma localização é uma localidade de um determinado país. De modo a povoar estas duas coleções, o grupo optou por utilizar um *script* de povoamento a partir de uma base de dados *online* de acesso livre, permitindo apresentar um cenário próximo do real em termos de quantidade de registos.

Quanto às categorias e especializações, estas constituem as áreas de trabalho e serviços mais concretos disponibilizados. A título de exemplo, uma possível categoria pode ser ”Ensino” e uma especialização associada a esta seria ”Ensino de História Romana”. É importante destacar que, tendo em conta a possibilidade de sugestão de quer categorias, quer especializações, estas possuem um campo indicando a sua visibilidade pelos utilizadores, podendo ser alterados pela administração.

A coleção dos utilizadores é um pouco mais complexa, devido ao maior número de ações a esta associadas. O campo de nível de acesso serve como máscara para indicar se o utilizador se trata de um solicitador, prestador ou administrador. O aspeto importante são os *arrays* de identificadores de categorias, especializações e localizações, no caso deste se tratar de um prestador de serviços, permitindo a sua rápida e fácil associação e procura. Além disso, uma imagem do prestador é armazenada na forma de extensão e conteúdo, sendo este uma *string base64*. Esta estratégia, apesar de gerar um aumento próximo dos 33% nos dados [1], permite a exibição direta da imagem numa interface reativa, como o caso da desenvolvida.

Por fim, os serviços possuem no seu corpo indicação dos identificadores tanto do cliente como do prestador. Possuem, também, um estado, de forma a verificar quais as ações possíveis a executar sobre um determinado registo, bem como campos descritivos sobre data, hora, duração, entre outros.

É importante mencionar que, de forma a permitir avaliações direcionadas a cada parte envolvente no serviço, existem dois objetos com diferentes parâmetros para estas, gerando um valor de *karma* que depois é somado ao *karma* geral do prestador. Outro aspeto associado aos serviços é o orçamento, onde os utilizadores podem indicar valores que estão dispostos a pagar/receber até que a outra parte aceite ou decidam cancelar completamente a negociação. Além disso, como um solicitador pode anunciar uma necessidade urgente, foi definido um estado próprio para tal, bem como a não necessidade de inclusão de um prestador que, posteriormente, se pode associar ao serviço, criando uma cópia do mesmo com o seu identificador já associado e um novo estado de negociação.

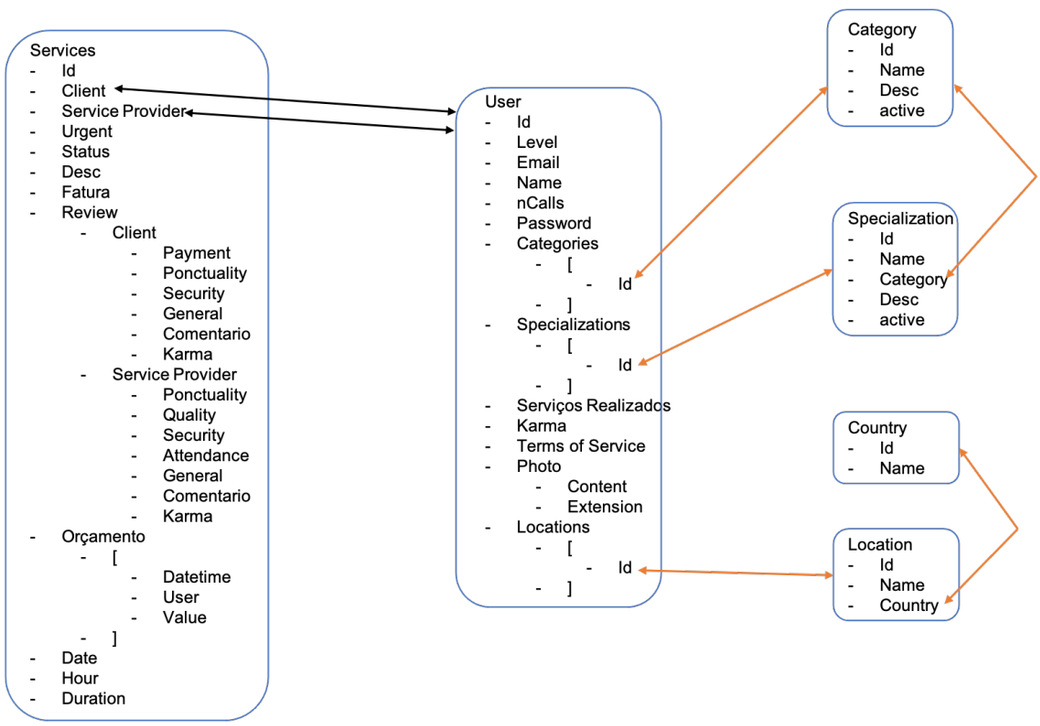

Figura 15 – Diagrama representante do modelo de dados da plataforma.

Assim, foi utilizada uma base de dados *No-SQL*, *MongoDB*, devido a este permitir o desenvolvimento de aplicações de forma rápida, lidando com tipos de dados variados e gerindo aplicações de forma eficiente, independentemente da escala [2]. Além disso, tendo em conta que objetos *MongoDB* mapeiam, diretamente, objetos, o uso desta base de dados permite a remoção da complexidade associada a *object-relational mapping* (*ORM*), necessária para traduzir objetos para tabelas relacionais [2]. Além do mais, a flexibilidade de *MongoDB* permite que o esquema de dados possa evoluir facilmente com as necessidades do negócio. Outro aspeto importante passa pela escalabilidade permitida através do uso de uma base de dados não relacional, escalando dentro e entre vários centros de dados distribuídos, gerindo os dados de uma forma não permitida por bases de dados relacionais. Assim, com o aumento dos volumes de dados e *throughput*, *MongoDB* funciona sem redução na *performance* e sem alterar a aplicação, algo que não aconteceria com *MySQL* [2].

3.2.2 *Backend*

Para o desenvolvimento do *backend* da aplicação, foi optado pelo uso de *NodeJS*, já que este se trata de um interpretador de *JavaScript* que permite construção de aplicações *web* fluídas e otimizadas. Além disso, permitem responder a diversos pedidos de forma assíncrona, ou seja, a vários utilizadores em simultâneo.

Além deste interpretador, a *API* de dados utiliza a *framework Express*, já que esta baseia-se na implementação de serviços *middleware* para responder a pedidos *HTTP*, como *GET*, *POST*, *PUT* e *DELETE* sobre rotas previamente definidas, com diferentes ações para a resposta aos pedidos. Por exemplo, por cada coleção existem diversas rotas, desde operações de consulta, criação e edição, até remoção. Assim, a utilização de uma *framework* que permita este nível de modularidade permite organizar a camada lógica da plataforma, facilitando a sua alteração ao longo do tempo, conforme necessário.

É, ainda, importante mencionar a utilização do módulo *Mongoose* para manipulação direta dos dados na base de dados através de controladores chamados nas diversas rotas, bem como o uso do *middleware* por este fornecido para simular *triggers*. Estes podem ser utilizados, por exemplo, para atualizar o número de serviços de um utilizador após este ser registado como finalizado, ou seja, receber as avaliações dos dois intervenientes.

De forma a proteger as rotas, já que nem todas podem ser acedidas por todos os utilizadores, foi utilizado um *middleware* de autenticação, verificando o nível de acesso do utilizador que fez o pedido através do cabeçalho. Assim, antes de analisar os dados, é possível recusar o pedido caso o nível não permita o acesso devido. Além disso, de forma semelhante, um módulo de validação foi utilizado para verificar tanto existência de campos nos pedidos, como para verificar se os seus valores eram válidos ou estavam de acordo com os formatos aceites na plataforma, como no caso da data. É importante destacar a possibilidade de uso de chaves de *API* para aceder a certos componentes dos dados, além do uso de *tokens* de utilizadores, e envio de *emails* para recuperação de *passwords* ou atribuição de uma chave, fazendo da plataforma algo mais complexo e próximo do real.

Aliado à implementação desta, todas as rotas foram documentadas, utilizando *Swagger*, uma *framework* para gerar documentação e permitir executar as várias rotas. Simultaneamente, a *Swagger* permite exemplificar resultados esperados por cada rota.

3.2.3 *Frontend*

Para a construção do *frontend*, ou seja, a interface utilizada pelos utilizadores para interagir com a plataforma, foi utilizada a *framework VueJS*, devido a ser uma ferramenta que permite a construção de interfaces reativas com um grau de acessibilidade bastante alto, bem como um crescendo na utilização por parte de desenvolvedores [3]. De forma a complementar esta ferramenta, foi utilizado o módulo *Vuetify*, trazendo diversas opções de estilo com uso simples e manipulação de dados eficiente. É, também, importante mencionar o uso de um *Linter*, ou seja, uma ferramenta de verificação de código com base num determinado conjunto de regras sobre identação, uso de determinadas ações, como *logging* na consola do *browser*, entre outros aspetos. Desta forma, apesar de existirem vários elementos a desenvolver a interface, o código acaba por seguir um certo padrão, facilitando a sua manutenção.

Além disso, devido à existência de rotas protegidas pelo uso de *tokens*, foi tirado proveito da existência de *stores* do *Vue*, armazenando informação como o nível de acesso, nome e *token* do utilizador aquando do seu *login*. Assim, tornou-se possível fazer um *render* condicional dos vários elementos *HTML* com base no nível, bem como restringir rotas utilizando este. Ademais, permite enviar pedidos diretamente ao *backend*, associando o *token* e nível aos mesmos, simplificando a tarefa de autenticação e permitindo verificar a veracidade destas ações.

3.2.4 *Deployment*

Como mencionado anteriormente, a aplicação desenvolvida é composta por um servidor de *backend*, um *frontend* e uma base de dados *MongoDB*. Assim, de forma a tornar a plataforma portátil e de fácil *deployment*, foram utilizadas as ferramentas *docker*, mais especificamente *docker-compose*, aliados ao isolamento dos diferentes componentes com redes do tipo *bridge*, conectando os serviços entre si, como se vê na Figura 16. Além disso, a arquitetura representada na Figura 16 descreve quer o ambiente de desenvolvimento, quer o ambiente de produção, ambos usando a ferramenta *PM2* para monitorização, mas diferindo em certos aspetos. A versão de desenvolvimento não possui um *Nginx reverse proxy*, usando o comando *npm run serve* do *VueJS* para criar um servidor. Sendo esta a razão pela qual que este não está apto a ser usado em produção para servir os ficheiros estáticos da interface e os serviços estão expostos nas portas locais da máquina onde correm. Já a versão de produção tem como ponto de entrada o *Nginx*, funcionando como *reverse proxy*, ou seja, redireccionando os pedidos para a *API* ou para a interface. Além disso, os ficheiros da interface gerados pelo comando *npm run build* são depois servidos por um *proxy Nginx* distinto.

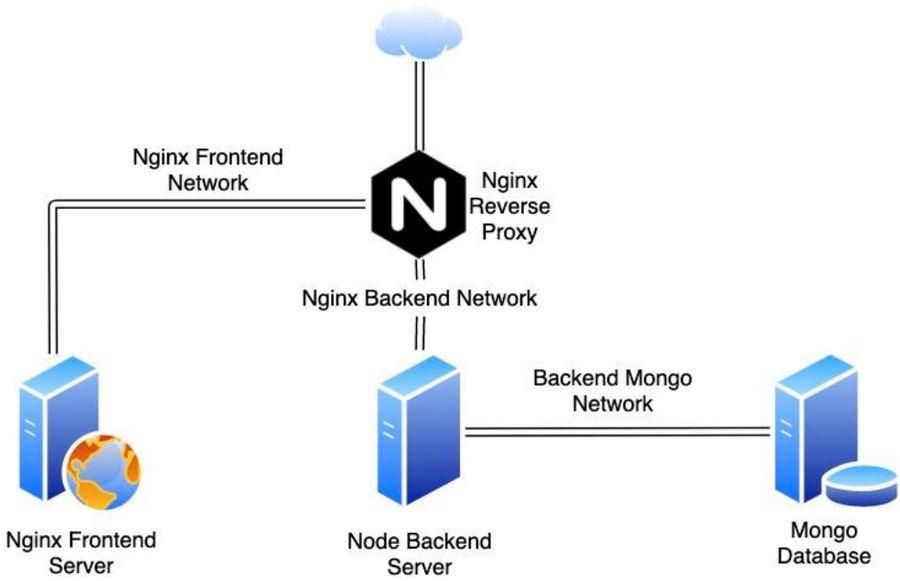

Figura 16 – Diagrama de arquitetura de sistema da plataforma após *deployment*.

Quanto ao *Nginx reverse proxy*, este redirecciona para o *backend* os pedidos que contenham */v1/*, com reescrita do *URI* enviado, assegurando-se que este */v1/* também é enviado para o *URI* da *API*. É, também, importante mencionar o caso especial da documentação *Swagger*, enviando apenas *docs.yaml* ao invés de juntar a versão do *backend*. Os restantes pedidos são, assim, enviados para o *frontend* através de uma regra genérica.

Em termos concretos, o *backend* faz *build* da imagem a partir de um *Dockerfile* criado especificamente para o mesmo, sendo possível escolher dois *targets* distintos, *dev* e *prod*. Quanto ao primeiro, a aplicação vigia os ficheiros de código e, após 60 segundos, efetua um *restart* do servidor para facilitar o desenvolvimento. Quanto ao segundo caso, apenas é efetuada uma conexão à ferramenta *PM2*, algo que também é efetuado na versão de desenvolvimento. Já o *frontend* possui, também, um ficheiro *Dockerfile* com os mesmos dois *targets*, diferindo apenas no segundo, sendo que este efetua um *build* do servidor *VueJS* e serve os ficheiros pedidos através de um *Nginx Web Server*. Por fim, a base de dados utilizada na aplicação é iniciada utilizando a imagem do repositório público *Bitnami*, facilitando algumas configurações, juntamente com um volume local para persistir a informação criada. Note-se que, de modo a efetuar uma prova de conceito, este volume encontra-se povoado com dados fictícios.

Tendo em conta a simplicidade da arquitetura desenvolvida, a adição de componentes ou *scaling* dos já existentes poderia ser necessária com o crescimento da plataforma. Assim, algumas possíveis adições passariam por adicionar mais métricas ao *PM2* na própria aplicação, permitindo seguir com mais precisão certos aspetos do seu funcionamento e, consequentemente, auxiliar na tomada de decisões. Além do mais, a ocorrência de falhas em aplicações em produção é possível, sendo causadas por problemas ou por períodos de manutenção, levando à necessidade de redundância de bases de dados e dos serviços de *frontend* e *backend*. Para tal, o *MongoDB* seria transformado num *replicaSet* e a *API* e interface teriam que ser escalados com auxílio de um *load balancer* no meio de cada camada. Apesar da qualidade das métricas retiradas através de uma ferramenta de monitorização como o *PM2*, seria importante fazer o mesmo para os serviços. Desta forma, ferramentas como *Prometheus* e *Grafana* poderiam ser adicionados à arquitetura para aglomerar as diversas métricas e *Fluentd* para agregar os *logs* gerados. No que toca à persistência de dados, teria que ser definido um esquema de *backups* para uma máquina diferente da máquina onde os dados estão armazenados ou uso de *clouds*, como *AWS S3 Buckets, Google cloud*, entre outras. Finalmente, a definição de regras de *firewall* restritas para os fluxos expectáveis de pacotes gerados pela aplicação seria necessária.

4 Gestão de Recursos

Ao longo deste capítulo são apresentadas as equipas de trabalho, bem como as ferramentas utilizadas para gestão das tarefas repartidas entre os vários elementos.

4.1 Equipas de Trabalho

No que toca às equipas, de forma a desenvolver um produto com qualidade, os elementos do grupo foram divididos com base nas cargas de trabalho de cada componente, como se pode ver em seguida.

- **Backend**: Nuno Rei, Paulo Barbosa
- **Frontend**: Fábio Senra, Jaime Leite, João Bernardo
- **Full-stack**: João Pimentel
- **Testes de Software**: Pedro Gonçalves, Jaime Leite
- **Deployment**: Paulo Bento

Note-se que, de forma a permitir uma maior facilidade de interligação entre as camadas da aplicação, um dos elementos, João Pimentel, foi colocado numa posição de *full-stack developer*.

4.2 Reuniões

Além desta divisão, a equipa realizou reuniões semanais com o seu orientador, o Professor José Carlos Ramalho, mostrando progresso à medida que este era feito. Aliado a estas reuniões, foram realizadas reuniões apenas entre os membros da equipa, pelo menos uma vez por semana, existindo sempre mais que um canal de comunicação sempre em funcionamento para o caso de serem necessárias alterações ou efetuar decisões sobre aspetos importantes.

No que diz respeito à componente de negócio, foram efetuadas diversas apresentações ao longo do período de desenvolvimento para os docentes, bem como apresentações e discussões regulares entre os alunos.

4.3 Tarefas

Quanto à gestão das tarefas entre cada equipa, foi utilizada uma folha de cálculo partilhada com múltiplos campos. Assim, foi possível testar aspetos chave de cada componente, bem como gerir, de forma fácil as tarefas realizadas, o que poderia ser melhorado e o que era necessário implementar. Além disso, por cada tarefa, um ou mais intervenientes foram associados, com base na forma como contribuíram, ou seja, se a implementaram, testaram, entre outras situações.

5 Conclusões e Trabalho Futuro

Ao longo do presente relatório encontra-se representado o resultado do trabalho prático da unidade curricular de Projeto em Engenharia Informática. Neste contexto foram abordados os passos referentes ao levantamento de requisitos, desenvolvimento e implementação de vários métodos que permitiram construir um produto viável e atrativo financeiramente para facilitar a requisição de serviços.

Desta forma, foi possível construir uma plataforma que cumpre todos os requisitos propostos a implementar nesta fase, tendo todos estes sido testados e validados pela equipa de testes. No entanto, se ocorresse um lançamento da aplicação no mercado, atualizações periódicas seriam necessárias para resolução de dependências, *bugs*, melhorias ou outras situações semelhantes. Além disso, apesar da funcionalidade completa do *Servicify*, algumas melhorias gráficas poderiam ser importantes para tornar a plataforma mais apelativa ao público em geral, bem como a simplificação de algumas ações. Ademais, seria necessária a criação de uma empresa, seguido de uma auditoria externa para confirmar o funcionamento do produto sem problemas. Ainda, funcionalidades extra como notificações, uso de *sockets* para tornar as negociações mais próximas de tempo real, possibilidade de *upgrade* de conta temporário, contas *premium* para clientes e permitir pagamentos entre clientes e prestadores diretamente na aplicação seriam interessantes. Por fim, o uso de *HTTPS* para garantir um nível extra de segurança, desenvolver uma interface multi-língua e uma aplicação *mobile* seriam fatores de sucesso. Porém, seria necessário garantir viabilidade da plataforma, quer por via de lançamento para público geral com obtenção de adesão significativa, quer por via de investimento externo.

Em suma, o desenvolvimento deste projeto permitiu pensar e desenvolver uma plataforma de raiz, com base em *feedback* do público alvo, tornando-a atrativa e com perspetivas de sucesso. Assim, o grupo dá a experiência como um êxito, não obstante da possibilidade de melhorias para obter uma maior adesão no mercado.

Referências

[1] What is the space overhead of base64 encoding? https://lemire.me/blog/2019/01/30/what-is-the-space-overhead-of-base64-encoding/. Acessed: 2020-12-10.

[2] Mongodb vs mysql. https://www.mongodb.com/compare/mongodb-mysql. Acessed: 2020-12-10.

[3] Reasons why vue.js is getting more traction every month. https://www.monterail.com/blog/reasons-why-vuejs-is-popular/. Acessed: 2020-12-10.

6 Anexos

6.1 Formulário Utilizado no Levantamento de Requisitos

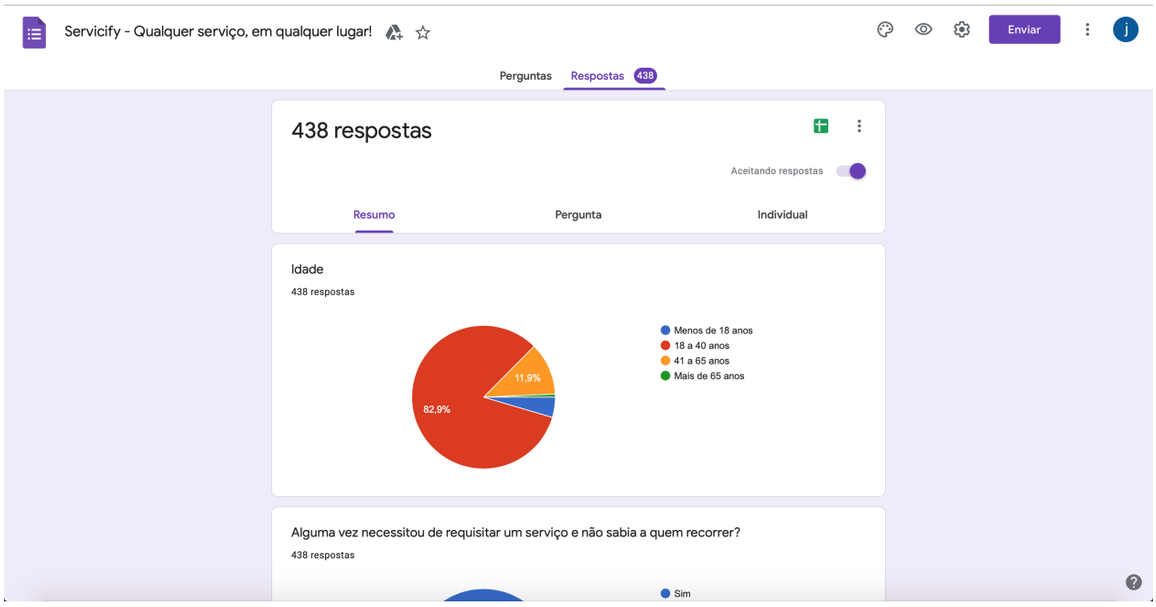

Figura 17 – Número total de respostas obtidas ao questionário e gráfico de idades dos inquiridos.

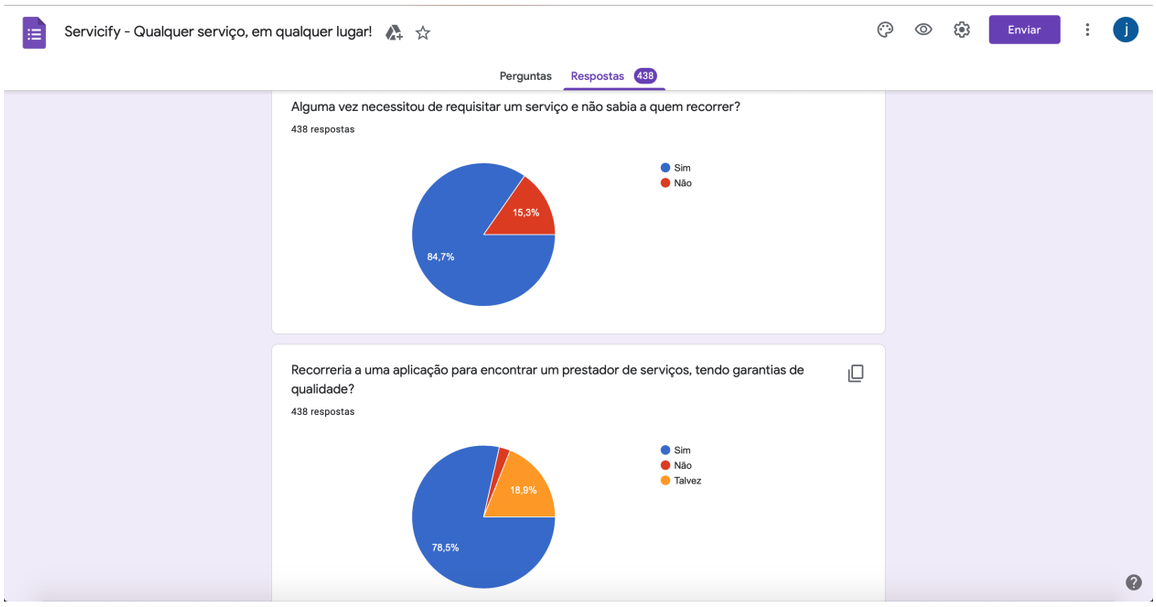

Figura 18 – Gráficos de respostas do formulário sobre necessidade de serviço e necessidade de aplicação.

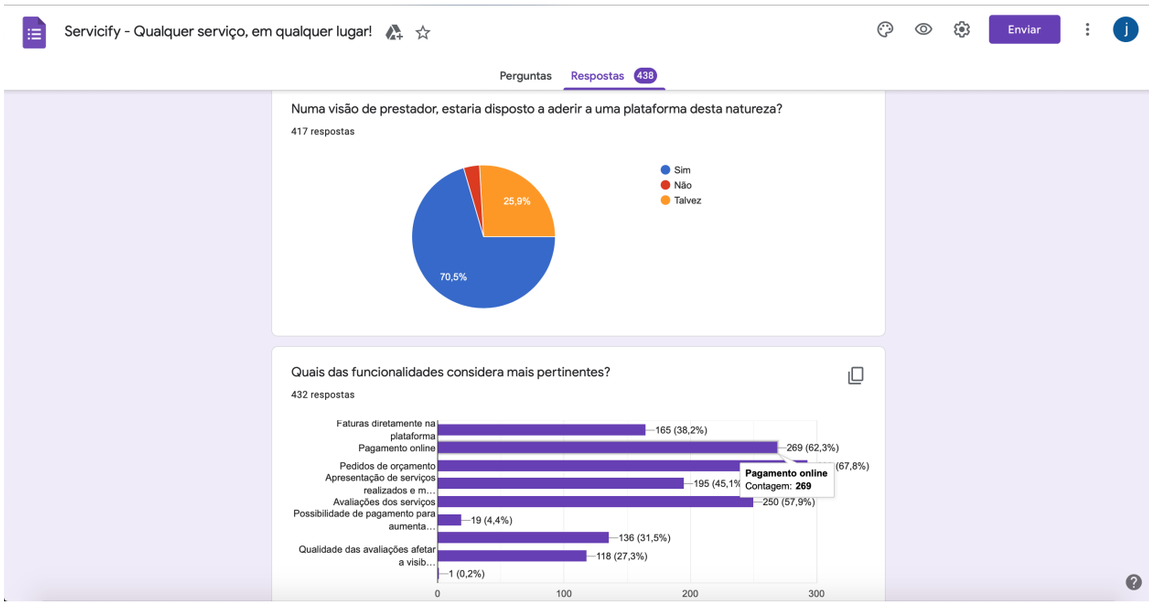

Figura 19 – Gráficos de respostas do formulário sobre adesão de prestadores e funcionalidades principais.

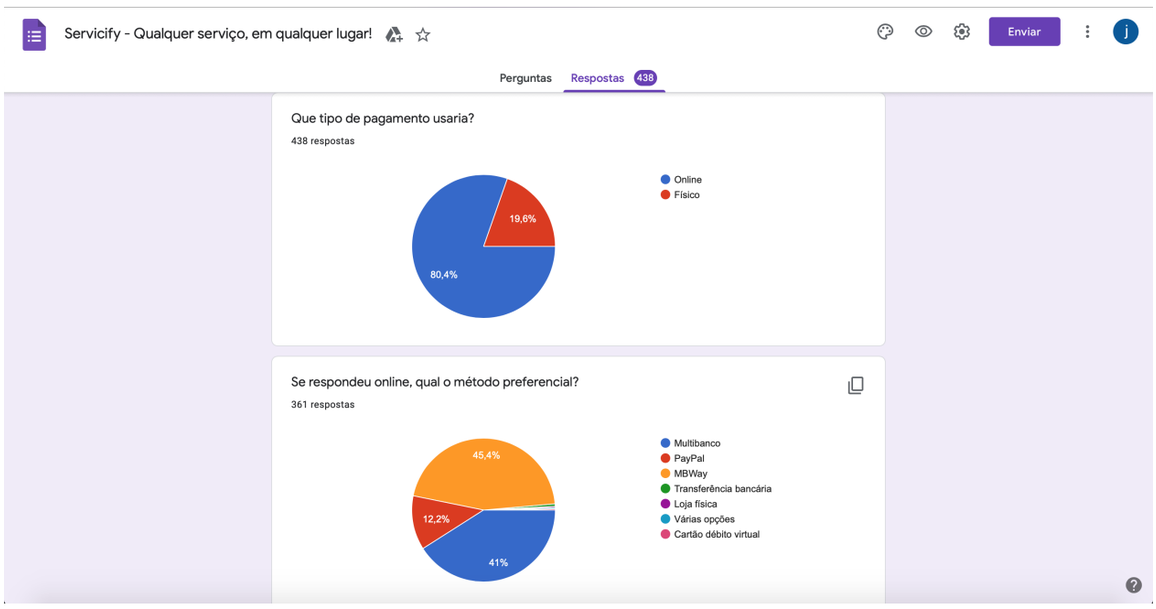

Figura 20 – Gráficos de respostas do formulário sobre tipos e métodos de pagamentos.

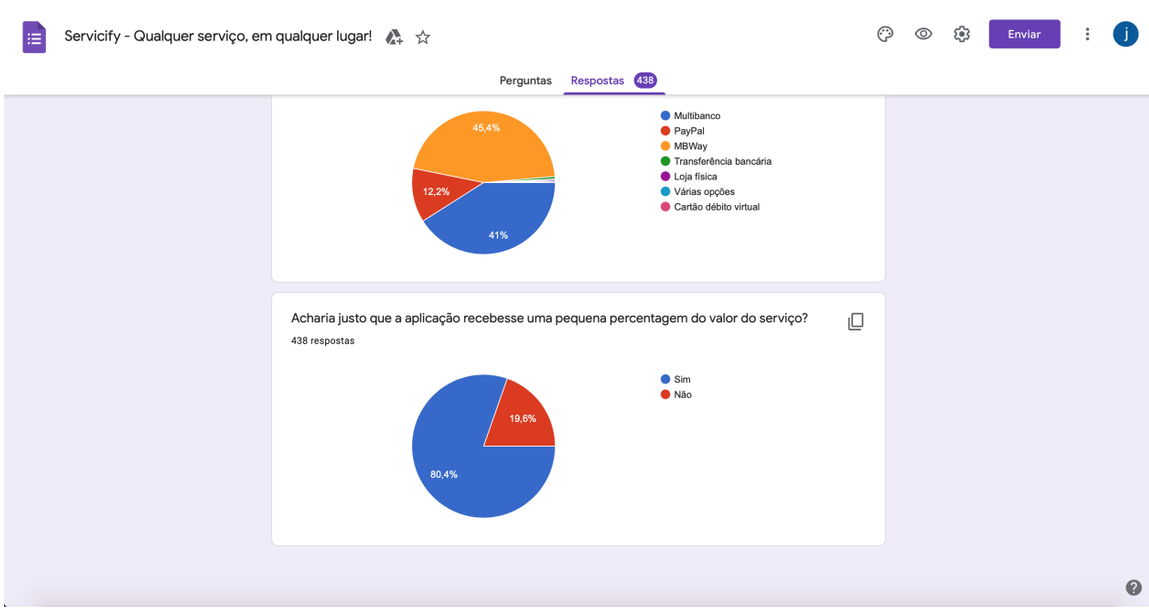

Figura 21 – Gráfico de respostas do formulário sobre taxa de utilização da aplicação.