



UNIVERSIDADE AUTÓNOMA DE LISBOA
LUÍS DE CAMÕES

DEPARTAMENTO DE CIÊNCIAS E TECNOLOGIAS LICENCIATURA
EM ENGENHARIA INFORMÁTICA

(APLICAÇÃO PARA GESTÃO DE RECURSOS HUMANOS)

Relatório de Projeto para a obtenção do grau de
Licenciado em Engenharia Informática

Autor/a: Diogo Fernandes de Mendonça Furtado,
João Alexandre Nunes Cotrim,
Pedro Maria Portugal e Castro Guilherme,
Vyacheslav Stafiyuk

Orientador/a: Dr. Gonçalo Valadão

Número do/a candidato/a: 30003503, 30001507, 30003419, 30001035

Julho de 2021

Lisboa

(Página em Branco)

Dedicatória

Diogo Furtado: Dedico este projeto a minha família e amigos que ajudaram a chegar a este ponto, com uma menção especial ao meu pai Henrique João Rodrigues de Mendonça Furtado, a minha mãe Adélia Cristina Teixeira Fernandes Furtado, e avô Amadeu Teixeira Fernandes, que sem os seus apoios não seria possível acabar esta jornada na Licenciatura em Engenharia Informática.

Vyacheslav Stafiyuk: Dedico este projeto a minha família e aos meus amigos que me apoiaram neste percurso, dando uma menção especial ao meu pai Viktor Stafiyuk, a minha mãe Nadiya Stafiyuk e a minha irmã Sónia Stafiyuk, sem o apoio dos quais não seria possível realizar a Licenciatura em Engenharia Informática.

Pedro Guilherme: Dedico este projeto aos meus pais e irmãos e amigos que me apoiaram neste percurso, dando especial atenção aos meus pais, Jorge Manuel Costa Guilherme e Teresa Maria Perestrello de França de Portugal e Castro, que me proporcionaram o curso. Sem eles não teria sido possível realizar esta Licenciatura em Engenharia Informática.

João Cotrim: Dedico este projeto principalmente aos meus pais e ao meu irmão que me apoiaram neste percurso e que estiveram presentes nos momentos mais difíceis. Aos meus pais, por terem me proporcionado uma educação de qualidade e por me terem transmitido valores de trabalho que levarei para a vida. Ao meu irmão por ter dado força de continuar a trabalhar nos momentos mais difíceis. Quero agradecer aos meus colegas de grupo que cresceram academicamente comigo como estudantes e trabalhadores e que concretizaram este projeto.

Agradecimentos

Este relatório foi desenvolvido na área de Engenharia Informática (EI), do departamento de Ciências e Tecnologias (DCT), da Universidade Autónoma de Lisboa (UAL). Este trabalho foi realizado com o apoio

Agradecemos ao professor Gonçalo Valadão, orientador deste projeto pelo todo o tempo disponibilizado ao esclarecimento de dúvidas e todos os professores que nos ajudaram a concluir esta licenciatura de Engenharia Informática pois sem eles não tínhamos conseguido fazer este projeto final.

Por fim, gostaríamos de fazer um especial agradecimento á nossa família, que nos ajudaram muito com o apoio constante durante toda esta licenciatura incluindo este projeto.

Epígrafe

“Medir o progresso da programação por linhas de código é como medir o progresso da construção de aeronaves em termos de peso.” - Bill Gates

“Os primeiros 90% do código representam os primeiros 10% do tempo de desenvolvimento. Os 10% restantes do código é para mostrar aos outros os 90% do tempo de desenvolvimento.” - Tom Cargill

“Há duas formas de construir um projeto de software: Uma maneira de fazer isso deve ser tão simples que, obviamente, não deixem deficiências, e a outra forma é a de torná-lo tão complicado que não percebiam as evidentes deficiências. O primeiro método é muito mais difícil.” - CAR Hoare

Resumo (Português)

Este trabalho tem como objetivo a criação de um site que para a gestão de recursos humanos de uma empresa na área das limpezas domésticas chamada Cillit Bang, feita em JavaScript, a primeira página é onde o cliente pode se registrar no nosso site e fazer o *login*. Após fazer o *login* em será direcionado para a página de cliente onde poderá fazer as suas marcações, verificar as mesmas e também poderá fazer o *feedback* da limpeza. Os trabalhadores quando entram no site poderão fazer *login* para a sua página de trabalhador em que podem ver as suas marcações e o *feedback* dos clientes. Os administradores podem também fazer *login* para a página de administradores que permite criar equipas de trabalhadores em que podem associar/desassociar a um determinado projeto que está ligado a um cliente em específico, podendo também utilizar o site para registrar novos administradores e trabalhadores, ver os seus projetos/marcações e ver os clientes registados no seu site.

Inicialmente pensamos em ter só uma página onde se pode fazer o *login* e dependendo das credenciais o utilizador iria ser redirecionado para a página certa, mas não conseguindo arranjar uma solução para isso decidimos fazer 3 páginas de *login* uma para o cliente, administrador e trabalhador

Para a criação do nosso programa utilizamos o Node.js que já tínhamos usado em Desenvolvimento Web no semestre passado o que nos facilitou bastante na realização deste projeto.

Utilizamos vários módulos para a criação desta aplicação *web* como passport para a autenticação de utilizadores, MongoDB é onde está a nossa base de dados, connect-flash para guardar mensagens numa sessão, express para criar a web application, express-session para criar uma sessão middleware com varias opções, bcryptjs para ajudar a fazer hash na password, nodemailer para enviar um email de confirmação quando um utilizador se regista, jsonwebtoken para termos um método autenticação e passport-local que é uma estratégia local do modulo passport.

Palavras-chave: JavaScript; Login; Cliente; Trabalhador; Administrador, MongoDB

Abstract (Inglês)

This assignment aims to create a website that for the human resource management of a company in the field of domestic cleaning called Cillit Bang, made in JavaScript, the first page is where the customer can register on our site and login. After logging in you will be directed to the customer page where you can make your appointments, check them and you can also write the feedback of cleaning. Workers when they enter the site will be able to log in to their worker page where they can view their appointments and customer feedback. Admins can also login to the admin page which allows them to create teams of workers where they can associate/disassociate with a particular project that is linked to a specific client, they can also use the site to register new admins and workers, view their projects/bookings and view the clients registered on their site.

Initially we thought of having only one page where you can login and depending on your credentials the user would be redirected to the right page, but not being able to find a solution for that we decided to make 3 login pages one for the client, administrator and worker.

To create our program, we used Node.js that we had already used in Web Development last semester which made this project a lot easier.

We used several modules to create this web application such as passport for user authentication, MongoDB is where our database is, connect-flash to store messages in a session, express to create the web application, express-session to create a middleware session with several options, bcryptjs, to help hash the password, nodemailer to send a confirmation email when a user registers, jsonwebtoken to have an authentication method and passport-local which is a local strategy for the passport module.

Keywords: JavaScript; Login; Client; Administrator; Worker, MongoDB

Resumée (Francês)

Ce travail vise à créer un site web qui pour la gestion des ressources humaines d'une entreprise dans le domaine du nettoyage domestique appelé Cillit Bang, fait en JavaScript, la première page est où le client peut s'inscrire sur notre site et se connecter. Après avoir ouvert une session, vous serez dirigé vers la page client où vous pourrez prendre vos rendez-vous, les vérifier et donner votre avis sur le nettoyage. Lorsqu'ils entreront sur le site, les travailleurs pourront se connecter à leur page de travail où ils pourront consulter leurs rendez-vous et les commentaires des clients. Les administrateurs peuvent également se connecter à la page d'administration qui leur permet de créer des équipes de travailleurs où ils peuvent s'associer/désassocier à un projet particulier lié à un client spécifique, ils peuvent également utiliser le site pour enregistrer de nouveaux administrateurs et travailleurs, visualiser leurs projets/réservations et visualiser les clients enregistrés sur leur site.

Au départ, nous pensions n'avoir qu'une seule page de connexion et, en fonction des informations d'identification, l'utilisateur serait redirigé vers la bonne page, mais comme nous n'avons pas trouvé de solution, nous avons décidé de créer 3 pages de connexion : une pour le client, une pour l'administrateur et une pour le travailleur.

Pour la création de notre programme, nous avons utilisé Node.js que nous avons déjà utilisé dans le développement Web du semestre dernier, ce qui a rendu ce projet beaucoup plus facile.

Nous avons utilisé plusieurs modules pour créer cette application web comme passport pour l'authentification des utilisateurs, MongoDB où se trouve notre base de données, connect-flash pour stocker les messages dans une session, express pour créer l'application web, express-session pour créer une session middleware avec plusieurs options, bcryptjs pour aider à hacher le mot de passe, nodemailer pour envoyer un email de confirmation quand un utilisateur s'enregistre, jsonwebtoken pour avoir une méthode d'authentification et passport-local qui est une stratégie locale du module passport

Mots-clés: JavaScript; Connexion; Client; Administrateur; Travailleur; MongoDB.

Zusammenfassung (Alemão)

Diese Arbeit zielt darauf ab, eine Website zu erstellen, die für die Personalverwaltung eines Unternehmens im Bereich der Haushaltsreinigung namens Cillit Bang, in JavaScript gemacht, ist die erste Seite, wo der Kunde auf unserer Website registrieren und anmelden können. Nach dem Einloggen werden Sie auf die Kundenseite weitergeleitet, wo Sie Ihre Termine vereinbaren, überprüfen und auch eine Rückmeldung zur Reinigung geben können. Arbeiter können sich beim Betreten der Website auf ihrer Arbeiterseite einloggen, wo sie ihre Termine und Kundenfeedback einsehen können. Admins können sich auch auf der Admin-Seite anmelden, die es ihnen ermöglicht, Teams von Arbeitern zu erstellen, in denen sie sich mit einem bestimmten Projekt, das mit einem bestimmten Kunden verknüpft ist, verbinden/entkoppeln können, sie können die Seite auch nutzen, um neue Admins und Arbeiter zu registrieren, ihre Projekte/Buchungen zu sehen und die auf ihrer Seite registrierten Kunden zu sehen.

Ursprünglich dachten wir daran, nur eine Seite zu haben, auf der man sich anmelden kann, und je nach den Anmeldeinformationen würde der Benutzer auf die richtige Seite weitergeleitet werden, aber da wir keine Lösung dafür finden konnten, entschieden wir uns, 3 Anmeldeseiten zu erstellen, eine für den Kunden, den Administrator und den Arbeiter

Für die Erstellung unseres Programms haben wir Node.js verwendet, das wir bereits im letzten Semester in Web Development eingesetzt hatten, was dieses Projekt sehr erleichtert hat.

Wir haben mehrere Module verwendet, um diese Webanwendung zu erstellen, wie z. B. passport für die Benutzerauthentifizierung, MongoDB, wo sich unsere Datenbank befindet, connect-flash, um Nachrichten in einer Sitzung zu speichern, express, um die Webanwendung zu erstellen, express-session, um eine Middleware-Sitzung mit mehreren Optionen zu erstellen, bcryptjs, um den Hash des Passworts zu unterstützen, nodemailer, um eine Bestätigungs-E-Mail zu senden, wenn sich ein Benutzer registriert, jsonwebtoken, um eine Authentifizierungsmethode zu haben, und passport-local, was eine lokale Strategie des passport-Moduls ist.

Schlüsselwörter: JavaScript; Anmelden; Kunden; Arbeiter; Administrator; MongoDB.

Реферат (Ucrainiano)

Ця робота спрямована на створення веб-сайту для управління людськими ресурсами компанії в галузі побутового прибирання під назвою Cillit Bang, виконаного на JavaScript, перша сторінка - це те, де клієнт може зареєструватися на нашому веб-сайті та увійти. Після входу вас перенаправлять на сторінку клієнта, де ви можете назначати прибирання, перевіряти їх, а також надати відгук про них. Працівники, зайшовши на сайт, зможуть увійти на свою сторінку працівника, де вони зможуть побачити свої назначення а також оставити відгук. Адміністратори також можуть увійти на сторінку адміністраторів, що дозволяє створювати команди працівників, де вони можуть приєднатись / від'єднатись від певного проекту, пов'язаного з певним клієнтом, а також можуть використовувати сайт для реєстрації нових адміністраторів та працівників а також проектів / закладок, і побачити зареєстрованих клієнтів на веб-сайті.

Спочатку ми думали мати лише одну сторінку, куди ви можете увійти, і залежно від облікових даних користувач буде перенаправлений на потрібну сторінку, але не маючи можливості знайти рішення для цього, ми вирішили зробити 3 сторінки входу для клієнта, адміністратора і робітник

Для створення нашої програми ми використали Node.js, який ми вже використовували у веб-розробці минулого семестру, що значно полегшило цей проект.

Ми використовуємо кілька модулів для створення цього веб-додатку як passport для автентифікації користувачів, MongoDB - це наша база даних, connect-flash для збереження повідомлень у сеансі, express для створення веб-програми, express-session для створення проміжного програмного забезпечення сеансу з декількома options, bcryptjs для допомоги в хешуванні пароля, nodemailer для надсилання електронного листа з підтвердженням, коли користувач реєструється, jsonwebtoken для використання методу автентифікації та passport-local, що є локальною стратегією модуля passport.

Ключові слова: JavaScript; увійти; клієнт; адміністратор; робітник; MongoDB.

Índice

Dedicatória	3
Agradecimentos	4
Epígrafe	5
Resumo (Português)	6
Abstract (Inglês)	7
Resumée (Francês)	8
Zusammenfassung (Alemão)	9
Резюме (Ucraniano)	10
Índice	11
Lista de Fotografias/Ilustrações	14
Lista de Siglas e Acrónimos	15
1. Introdução.....	16
2. Entidades Envolvidas.....	17
2.0. UAL	17
3. Ferramentas utilizadas para desenvolvimento do projeto	18
3.1. JavaScript	18
3.2. HTML.....	18
3.3. CSS	19
3.4. Bootstrap.....	20
3.5. Node.js	21
3.6. Express	21
3.7. Express-session	22
3.8. MongoDB	22
3.9. Mongoose	23
3.10. Passport	23

3.11.	Passport-local	24
3.12.	Bcrypt	24
3.13.	JQuery	25
3.14.	Ajax	26
3.15.	JSON	26
3.16.	JSON web token	27
3.17.	Nodemailer	28
3.18.	Connect-flash	28
3.19.	Chart.js	30
3.20.	Canvas	31
4.	Nome do primeiro capítulo (Desenvolvimento)	31
4.1.	MongoDB Schemas	32
4.1.1.	User (Cliente)	33
4.1.2.	Trabalhador	34
4.1.3.	Administrador	35
4.1.4.	Marcação (Marcação da Limpeza)	36
4.1.5.	Equipas	37
4.1.6.	Feedback	38
4.2.	Funcionalidades das páginas	39
4.2.1.	Login e Registo do cliente	39
4.2.2.	Login e Registo do trabalhador	43
4.1.1.	Login e Registo do administrador	46
4.1.2.	Welcome Page	48
4.1.3.	Client Page	49
4.1.4.	Worker Page	51
4.1.5.	Página de Back-office (Página dos administradores)	52
4.1.6.	Dashboard	54

4.1.7. Clientes	60
4.1.7.1. Ver Cliente.....	62
4.1.7.2. Ver Projeto	64
4.1.7.3. Trabalhadores	66
4.1.7.3.1. Ver Equipa	67
4.1.7.3.2. Criar Equipa	68
4.1.7.3.3. Ver todos os Trabalhadores.....	70
4.1.7.4. Projetos / Marcações.....	71
4.1.7.4.1. Ver Projeto	72
Conclusões.....	75
Bibliografia.....	76

Lista de Fotografias/Ilustrações

Figura 1 - Coleções da base de dados	22
Figura 2 - Exemplo de password com encriptação.....	24
Figura 3 - Mensagem de Sucesso.....	29
Figura 4 - Mensagem de insucesso	29
Figura 5 – Total de trabalhadores.....	30
Figura 6 – Total de clientes.....	30
Figura 7 - Modelo usado para o cliente.....	33
Figura 8 - Modelo usado para o trabalhador	34
Figura 9 - Modelo usado para o Administrador	35
Figura 10 - Modelo usado para a Marcação	36
Figura 11- Modelo usado para a criação de equipas	37
Figura 12- Modelo usado para o Feedback	38
Figura 13- Formulário de registo do cliente.....	39
Figura 14- Campo de login do cliente.....	40
Figura 15 - Página de informação do email para repor a password	41
Figura 16- Campos para repor a password.....	42
Figura 17- Formulário de registo de um novo trabalhador.....	43
Figura 18- Campos de login do trabalhador.....	44
Figura 19- Campo para indicar o email do trabalhador.....	44
Figura 20- Campos para repor a password do trabalhador.....	45
Figura 21- Formulário de registo de administradores	46
Figura 22- Campo de login do administrador	47
Figura 23- Campo para indicar o email do administrador.....	47
Figura 24- Campos para repor a password do administrador.....	48
Figura 25- Imagem da página welcome	48
Figura 26- Menu de navegação da página welcome	49
Figura 27- Imagem da página de cliente	49
Figura 28- Secção de marcação da página do cliente.....	50
Figura 29- Barra de navegação do trabalhador	51
Figura 30- Área do trabalhador	51
Figura 31- Barra de navegação do administrador	53
Figura 32- Imagem do dashboard	54
Figura 33- Gráfico de avaliações	55
Figura 34- Gráfico total dos trabalhadores.....	56
Figura 35- Gráfico do total de clientes.....	57
Figura 36- Total de marcações.....	58
Figura 37- Gráfico de comparações	59
Figura 38- Grafico de comparações sobre administradores e quantidade de trabalhadores.....	60
Figura 39- Informações sobre os clientes.....	61
Figura 40- Informações do cliente	62
Figura 41- Manager associado ou não.....	63
Figura 42- Feedbacks dos clientes	63
Figura 43- Informações da marcação	64
Figura 44- Página dos trabalhadores do back office	66
Figura 45- Página de ver equipa.....	67
Figura 46- Página de criar equipa	68
Figura 47- Página de ver todos os trabalhadores.....	70
Figura 48- Página projetos/marcações	71
Figura 49- Página ver projeto.....	72
Figura 50- Página feedbacks só acedida pelos administradores.....	73
Figura 51- Página do user do administrador	74

Lista de Siglas e Acrónimos

API	Application Programming Interface
CERN	European Council for Nuclear Research
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ID	Identity Card
JSON	JavaScript Object Notation
ODM	Object Data Modeling
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
TLS	Transport Layer Security
UAL	Universidade Autónoma de Lisboa
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium
XML	Extensible Markup Language

1. Introdução

Este documento pretende especificar detalhadamente o que foi feito durante o desenvolvimento deste projeto de final de curso de Licenciatura em Engenharia Informática, na cadeira de Laboratório de Projeto. O trabalho após o seu desenvolvimento, permite fazer os registos de clientes, trabalhadores e administradores e os seus respetivos *logins* autenticados, os clientes conseguem fazer as marcações de limpezas, os trabalhadores podem ver na sua página os seus projetos e os administradores conseguem criar equipas de trabalhadores e avaliar os mesmos.

O objetivo do projeto é criar uma aplicação *web* para gestão de recursos humanos de uma empresa de limpezas domésticas em que seja possível a clientes fazer marcações de limpezas, essas marcações vão estar ligadas a uma equipa de trabalhadores que é criada por um administrador que pode avaliar o sucesso do grupo de trabalhadores de L1 a L10, os trabalhadores possuem a sua própria página de *login*

Este documento tem especificado todo o processo de criação da aplicação web incluindo as tecnologias utilizadas para a realização do mesmo

2. Entidades Envolvidas

2.0. UAL

A Universidade Autónoma de Lisboa (UAL) é um estabelecimento de ensino superior privado reconhecido nos termos legais pelo Ministério da tutela do ensino superior como instituição de interesse publico, está localizada em Lisboa no Palácio dos Condes do Redondo. Foi criada em 13 de dezembro de 1985 pela Cooperativa do Ensino Universitário (CEU) sendo responsável pela gestão económica e financeira da universidade, enquanto centro de criação, transmissão, crítica e difusão da cultura, ciência e tecnologia.

A UAL conta com uma gama de serviços de apoio na esfera didático-pedagógica, assim como no plano dos serviços de apoio, designadamente: acesso à alimentação em cantinas e bares, reprografia, apoio bibliográfico e informático, isenção ou redução de propinas, seguro escolar, atribuição de bolsas, apoios a estágios e gestão de carreiras ou atividades desportivas e culturais. [1]

A UAL oferece licenciaturas mestradas e doutoramentos em várias áreas. Algumas das áreas de licenciatura são:

- Engenharia Informática
- Psicologia
- Gestão do Desporto
- Informática de Gestão
- Gestão
- Arquitetura
- Direito
- Administração de Unidades de Saúde

3. Ferramentas utilizadas para desenvolvimento do projeto

3.1. JavaScript

JavaScript é a linguagem que usamos para fazer as funções e *scripts* e é uma das mais poderosas e mais utilizadas linguagens de programação, que permite implementar funcionalidades complexas em páginas *web*. E de bastante alto nível, com sintaxe dinâmica fraca e multiparadigma (protótipos, orientado a objeto, imperativo e, funcional). É utilizada principalmente em programação cliente-side também pode ser utilizada do lado do servidor através de ambientes como o node.js. Esta linguagem juntamente com HTML e CSS é uma das principais tecnologias utilizadas na *World Wide Web* sendo essencial aos aplicativos da *web* em que a maior parte dos sites utiliza. [2]

Esta linguagem foi implementada para que os *scripts* pudessem ser executados do lado do servidor e interagissem com o usuário sem a necessidade deste script passar pelo servidor, controlando o navegador, realizando comunicações assíncronas e alterando o conteúdo do documento exibido. Neste momento os mecanismos JavaScript estão incorporados em muitos outros tipos de *software* como bancos de dados e processadores de texto.[2][3]

O JavaScript foi criado por Brendan Eich e lançado em 1995, quando a internet ainda era bastante nova. Marc Andressen, o fundador do recém-lançado navegador Netscape queria aumentar a capacidade do navegador utilizando mais elementos dinâmicos, mas também queria que estes elementos estivessem disponíveis para novos programadores amadores, então teve a ideia de criar uma nova linguagem de programação que seja simples e dinâmica. Brendan Eish foi então contratado por Marc Andresseem para criar uma linguagem de *scripting* para navegadores. Tinham a estratégia de utilizar a linguagem de programação Java da empresa Sun Microsystems para conseguirem superar o Internet Explorer da Microsoft, então implementaram uma pequena linguagem para complementar o Java chamado de JavaScript. [2][3]

3.2. HTML

O HyperText Markup Language (HTML) é uma linguagem que utilizamos para fazer o a estrutura da aplicação *web*. É por meio deste tipo de linguagem consistindo de vários

elementos que o navegador interpreta o conteúdo em forma de código e o mostra na forma visual como vemos quando acedemos a um site.

O HyperText refere se a *links* que se conectam páginas *web* umas a outras independentemente se conecta um site a outro ou apenas uma página *web* a outra, que é um aspeto fundamental da *web*. A parte do markup server para anotar texto, imagens entre outros elementos como <head>, <title>, <body>, <header>, <footer>, <article>, <section>, <p>, <div>, , , <aside>, <audio>, <canvas>, <datalist>, <details>, <embed>, <nav>, <output>, <progress>, <video>, , , e são com estas marcações que se separa os elementos uns dos outros. [4]

O HTML foi criado em 1991, por Tim Berners-Lee, no European Council for Nuclear Research (CERN) que se inspirou na linguagem SGML a fim de desenvolver o que conhecemos hoje por HTML, foi dessa linguagem que foi herdado as inúmeras *tags*. É a linguagem mais usada para a criação de páginas *web* sendo a primeira linguagem mundial. Inicialmente surgiu como solução para interligar instituições de pesquisa próximas para compartilhar ficheiros e documentos com facilidade entre pesquisadores. [5]

O HTML é bastante fácil de utilizar, mas nem sempre foi assim antigamente era difícil de aprender pois eram muitos comandos para fazer coisas simples, agora com cada atualização fica mais simples de utilizar. [5]

3.3. CSS

O Cascading Style Sheets (CSS) é uma linguagem declarativa que controla a representação visual de uma página *web* separando esse conteúdo do site.

É constituída por camadas, controlando o layout a definição de cores, cor do texto, fundo, espaçamento entre parágrafos, criar tabelas entre outros, utiliza um seletor para seleccionar os elementos certos para aplicar um estilo ao site.

O CSS foi desenvolvido pelo *World Wide Web Consortium* (W3C) em 1996, pois o HTML não foi desenvolvido para ter *tags* que ajudam a formatar a página o que criava vários problemas para os desenvolvedores com *tags* como que foram introduzidas na versão 3.2 do HTML, o criava processos longos e dolorosos para reescrever o código pois tinha se que descrever a estilização na marcação HTML separadamente, todas as cores das fontes, alinhamentos entre outros.

Mas com o CSS é possível colocar toda a estética num ficheiro diferente, criando assim o estilo separado do código HTML, mas tendo que fazer a integração do ficheiro de CSS com o de HTML, mantendo assim os ficheiros muito mais fáceis de entender e organizar.

Só com um ficheiro de CSS reduz muito mais o código pois não é necessário repetir o código sempre que queremos utilizar o mesmo estilo e ainda aumentando as possibilidades de estética.[6] [7]

3.4. Bootstrap

BootStrap é das *frameworks* mais populares para o design de *front-end*, e foi utilizada neste projeto para a construção de interfaces modernas, responsivas e dinâmicas para o desenvolvimento profissional de páginas *web* e que recebeu uma nova atualização recentemente. Em abril de 2021, BootStrap foi considerado o décimo projeto mais bem-sucedido de sempre no Github.

É uma coleção de código CSS e JavaScript/jQuery grátis e *open-source* com o propósito de construir o *layout* de *websites* e aplicações *web*.

Nos seus primórdios, Bootstrap designava-se por Twitter Baseline e foi desenvolvido por Mark Otto e Jacob Thornton com o objetivo de encorajar a consistência entre ferramentas visto que, na altura, utilizava-se várias diferentes bibliotecas para o desenvolvimento de interfaces que por sua vez levariam a inconsistências e a um enorme fardo na manutenção. De acordo com o developer Mark Otto:

“Eu e um pequeníssimo grupo de desenvolvedores desenhámos e construímos uma nova ferramenta interna e vimos uma oportunidade para criar algo mais.

Durante esse processo, demos por nós construindo algo bastante mais substancial do que uma ferramenta interna. Meses depois, acabámos por criar uma versão inicial do Bootstrap como uma forma de documentar e partilhar padrões de design e ativos dentro da empresa.”

O objetivo principal de utilizar Bootstrap é a customização da página *web* ao gosto dos desenvolvedores, podendo escolher entre diferentes cores, tamanhos e *layouts* para o projeto.

Bootstrap fornece vários componentes JavaScript na forma de plugins jQuery, assim como elementos adicionais de interface do utilizador tais como *dialog boxes*, *tooltips* e *carousels*, sendo que alguns foram utilizados para este projeto.[8][9]

3.5. Node.js

Node.js é um *software* que utilizamos para conseguirmos construir aplicações *web* de código aberto, multiplataforma que permite a execução de códigos JavaScript e uma coleção de módulos que lidam com várias funcionalidades importantes fora de um navegador *web*.

Este *software* permite que os desenvolvedores corram comandos e scripts do lado do server para produzir uma página *web* dinâmica antes de ser enviada para o navegador do utilizador. Node.js representa um paradigma de JavaScript em todo o lado, que serve para unificar as linguagens de programação de aplicações *web* numa só linguagem em vez de linguagem diferentes para os *scripts* do lado do cliente e lado do server. O código é baseado na arquitetura *event-driven* capaz da entrada e saída assíncrona que é otimizado para ser corrido em tempo real.

O Node.js foi escrito por Ryan Dahl em 2009, criando este *software* pois os outros servidores *web* mais populares tinham possibilidades muito limitadas pois não conseguiam lidar com muitas conexões e bloqueava o processo de criação de código no caso de ligações em simultâneo. [10]

3.6. Express

É um módulo para o Node.js que serve de *back end* para o nosso site, o seu *software* é *open-source*, tendo também vários tutoriais, recursos e documentação para os utilizadores. Foi desenhado para construir aplicações web e APIs, sendo bastante rápido e facilitando o desenvolvimento de aplicações *back-end*. É dos módulos mais usados para *frameworks* do Node.js. Permite a desenvolvedores criar vários tipos de ferramentas do lado do server em JavaScript. [11]

As características do express são:

- Possuir um sistema de rotas completo;
- Possibilita o tratamento de exceções dentro da aplicação;
- Permite a integração de vários sistemas de *templates* que facilitam a criação de páginas web para as suas aplicações;
- Gerência diferentes requisições HTTP com seus mais diversos verbos;
- Feito para a criação rápida de aplicações utilizando um conjunto pequeno de arquivos e pastas. [12]

3.7. Express-session

É um módulo do Node.js que é usado para associar um pedido a outro pedido guardando dados entre pedidos de HTTP. Cookies e parâmetros de URL são ambos usados para transmitir dados entre cliente e servidor, mas tem um grande problema de segurança pois podem ser lidos no lado do cliente.

O cliente fica com um ID e faz todos os próximos pedidos usando esse ID. [13]

3.8. MongoDB

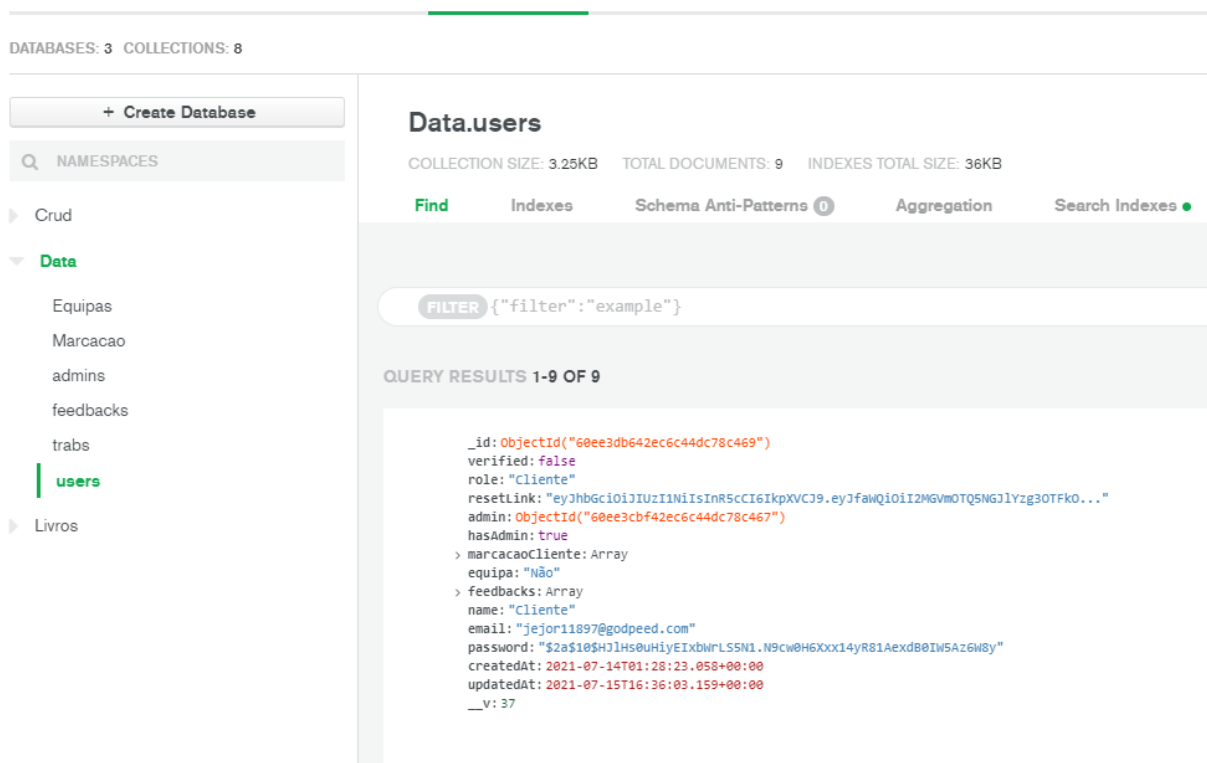


Figura 1 - Coleções da base de dados

Fonte: Printscreen tirado do das coleções da nossa base de dados

MongoDB é o *software* de base de dados que utilizamos para guardar as informações dos clientes, marcações, trabalhadores, administradores, equipas e *feedback* que está mostrado na Figura 1. É um *software* orientado a documentos livre, de código aberto e multiplataforma, escrito na linguagem C++. [14]

Este *software* é classificado por um programa de base de dados NoSQL open-source no formato Java Script Object Notation (JSON), ou seja, é diferente de uma base de dados relacional pois não necessita de ter tabelas e colunas criadas previamente, permitindo que um

documento represente toda a informação necessária, com todos os dados que queremos, no formato de um JSON, facilitando a performance das consultas realizadas. Assim podem existir valores simples como números, *strings*, datas, listas de objetos. Todos os documentos são agrupados em *collections* e o conjunto de *collections* forma uma base de dados. Tem bastante flexibilidade e uma excelente performance trazendo uma serie de possibilidades para a sua utilização. É uma base de dados sem regras específicas. [15] [14]

O MongoDB foi lançado em 2007 por uma empresa chamada 10gen, mas acabou por se tornar um banco open-source em 2009. [14]

3.9. Mongoose

Mongoose é uma biblioteca de Object Data Modeling (ODM) para MongoDB e Node.js e é materializado como um *Web Service*. É escrito em JavaScript e faz a gestão de relações entre dados, fornece validação de *Schema* e é utilizado para fazer a tradução entre objetos no código e a respetiva representação desses objetos na base de dados MongoDB, num ambiente assíncrono.

Como referido em cima, o MongoDB é uma base de dados NoSQL *schema-less*, o que significa que é possível guardar documentos JSON, e a sua estrutura pode variar, algo que não é possível em base de dados SQL, representando então uma vantagem ao utilizar NoSQL visto que acelera o desenvolvimento da aplicação e reduz a sua complexidade na produção das aplicações. [16]

3.10. Passport

Passport é um módulo do Node.js que utilizamos para autenticação de passwords do cliente, administrador e trabalhador, é um *middleware* do Node.js. Esta desenhado apenas para pedidos de autenticação não estando envolvido com o resto da aplicação incluindo assumir uma particular base de dados o que faz a integração bastante simples com o resto do código da aplicação.

A autenticação tem uma variedade de formas, normalmente é utilizado um nome de utilizador e uma *password*, mas este modulo, consegue usar outra informação como o email e *password*, ainda permite usar o *login* de redes sociais como Twitter, Facebook, Instagram ente outros, ao usar o fornecedor OAuth dessas redes sociais. Utilizando credenciais baseadas em *tokens*.

Depois de fazer *login* o passport vai manter seções de *login* persistentes, mas para isso funcionar vai ser necessário o utilizador autenticado ser serializado na sessão e de serializado quando for pedido [17]

Este módulo utiliza cerca de 520 estratégias que podem ser escolhidas pelo desenvolvedor pois cada aplicação necessita de métodos de autenticação diferentes. Estas estratégias podem ser instaladas como outros módulos individuais no Node.js sem dependências desnecessárias. [18]

3.11. Passport-local

Passport-local é uma estratégia do passport que o configura e é o módulo do None.js que utilizamos para autenticar e serializar os clientes, trabalhadores e administradores nas suas contas. Este modulo guarda localmente a *password* e email do utilizador na base de dados da nossa aplicação, não localmente no utilizador.

Normalmente é usada para autenticar com um nome de utilizador e a sua *password*, mas utilizamos para autenticar com email e *password* requerendo um *callback*, que aceita estas credenciais fornecendo um utilizador

Outra estratégia que podíamos ter utilizado é a password-jwt que utiliza JSON Web *Tokens* mas na nossa aplicação funcionava melhor a estratégia passport-local. [19]

3.12. Bcrypt

```
_id: ObjectId("60ee3cbf42ec6c44dc78c467")
verified: false
role: "Adminstrador"
resetLink: ""
> clients: Array
  firstName: "Admin"
  lastName: "One"
  email: "gisira3429@godpeed.com"
  password: "$2a$10$HF52mP6eOHHJvbXD0QOld.EMG8XHu51j8s1gdrbHZSEb6imzQBui"
  createdAt: 2021-07-14T01:24:15.268+00:00
  updatedAt: 2021-07-14T21:32:08.656+00:00
  __v: 15
```

Figura 2 - Exemplo de password com encriptação

Fonte: Printscreen tirado da base de dados no nosso site

Bcrypt é um método de criptografia implementado em diversas linguagens do tipo hash que usamos para encriptar as *passwords* do cliente, trabalhador e administrador como pode ver na Figura 2. É um dos métodos que apresenta mais segurança pois não tem fraquezas conhecidas, comparando com os outros métodos de criptografia como o método mais tradicional “crypt” que possuía restrições de espaço para guardar o hash e o salt gerado da senha criptografada, foi criado por David Mazières e é baseado na cifra de Blowfish.

Bcrypt é conhecido como um hash adaptativo pois consegue se adaptar as melhorias futuras de hardware, com esta característica consegue-se manter resistente a ataques do tipo “força bruta” [20]

3.13. JQuery

Jquery é uma pequena, rápida package de Javascript. Tem como por objetivo facilitar a manipulação dos documentos HTML, event handling, animações e o Ajax com a utilização de uma API que é interoperável com uma variedade de browsers. É considerada uma ferramenta antiga, considerando-se alternativas mais recentes, mas que permite oferecer uma combinação de versatilidade e extensibilidade aos programas. [21]

É um *software open-source* grátis que foi desenhado para simplificar a navegação de um documento, selecionando a informação dos DOM *elements* em Javascript. Esta tecnologia caracteriza-se por criar um novo estilo de programação ao juntar algoritmos com as estruturas de dados DOM.

JQuery fornece um paradigma para *event handling* que vai para além da seleção e manipulação dos elementos DOM. A designação do evento e a sua respetiva função *callback* são feitas num único passo, numa única localização do código. JQuery incorpora outras funcionalidades muito utilizadas de JavaScript nomeadamente esconder elementos, animações ao manipular propriedades CSS e algumas funções Ajax. [22]

Os princípios para desenvolver com jQuery são:

- A separação do JavaScript e HTML. Ao adicionar *event handlers* ao DOM utilizando JavaScript, encoraja os desenvolvedores em separar completamente código JavaScript de markup HTML.
- Redução de código e fácil entendimento.
- Eliminação de incompatibilidades entre browsers.
- Escalabilidade. [22]

3.14. Ajax

Asynchronous JavaScript and XML é um conjunto de técnicas de desenvolvimento *web* que utiliza inúmeras tecnologias *web* no lado cliente para a criação de aplicações *web* assíncronas.

Com Ajax as aplicações *web* conseguem enviar e receber informação do servidor de forma assíncrona sem interferir com o comportamento da página em questão. Ao dissociar a camada de troca de informação da camada de apresentação, é possível permitir que páginas *web* e, por extensão, aplicações *web*, alterarem dinamicamente o conteúdo de uma página sem ter a necessidade de recarregá-la. Na prática, as aplicações mais modernas utilizam comumente o JSON invés do XML.

No artigo que associou o termo Ajax ao conjunto de tecnologias *web*, escrito por Jesse James Garret, ele explicou que as seguintes tecnologias estão incorporadas:

- HTML e CSS para apresentação;
- O DOM para o display e a interação da informação;
- JSON ou XML para a troca de informação;
- Objeto XMLHttpRequest para comunicação assíncrona;
- JavaScript para juntar estas tecnologias juntas.

Desde então, inúmeras alterações ocorreram nestas tecnologias e na própria definição do termo Ajax. XML já não é necessário, recorrendo ao JavaScript Object Notation (JSON) como alternativa de formato para a troca de informação. JQuery inclui abstrações para o suporte na execução de pedidos Ajax, muitas delas utilizadas neste projeto.[23]

3.15. JSON

JavaScript Object Notation é um formato de troca de informação *lightweight* utilizado neste Projeto. É uma forma que nós humanos possamos ler e escrever linguagem *markup* e foi desenhada de forma a que seja fácil para máquinas analisarem e criarem.

JSON é baseado no JavaScript Programming Language Standard ECMA-262 3rd Edition – Dezembro de 1999 - e é um formato de texto que é completamente independente de

qualquer linguagem, mas utiliza convenções que são familiares a programadores da família de linguagens C nomeadamente, C, C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades são o que tornam JSON ideal para uma linguagem de troca de dados.

É construído sobre duas estruturas:

- Uma coleção de pares nome/valor.
- Uma lista de valores ordenados como um *array*, vetor, lista ou sequência.

Estas são estruturas de dados universais, das quais existem suporte para todas as linguagens de programação modernas. [24]

JSON imergiu pela necessidade de um protocolo cliente-servidor de comunicação *stateless*, em tempo real, sem a utilização de plugins como o Flash ou Java applets.

Uma das utilizações deste formato de texto é em JSON-RPC, ou seja, *Remote Procedure Call* construído sobre JSON, como alternativa a XML-RPC, também conhecido como SOAP. É um protocolo simples que define um conjunto de tipos de dados e respetivos comandos. Permite enviar notificações (informações do servidor que não requerem uma resposta) e múltiplas chamadas para o servidor que podem ser respondidas fora de ordem.

Outra utilização de JSON, necessária para a construção deste projeto, é em AJAX. Asynchronous JavaScript and JSON refere-se à metodologia Ajax, na criação de páginas *web* dinâmicas, que fornece a capacidade de uma página *web* fazer um pedido de informação após a página tiver sido carregada no browser. Tipicamente faz a renderização de nova informação do servidor em resposta a ações por parte dos utilizadores. [25]

3.16. JSON web token

JSON web token é um dos módulos do Node.js que utilizamos para autenticar o utilizador de quando enviamos um email para confirmar o a conta do mesmo. É um método que permite autenticar sem guardar nenhuma informação sobre o utilizador.

Quando o servidor gera um *token* declarando que o utilizador esta logado como administrador, trabalhador ou cliente ele é assinado pela chave privada para que depois possa ser verificada se o *token* é legítimo.

Estes *tokens* foram projetados para serem compactos e seguros para URL, especialmente para um login único. Quando um utilizador faz login com as credenciais corretas um JSON web *token* é devolvido e guardado localmente em vez de criar uma sessão no server e devolver uma cookie. [26]

3.17. Nodemailer

É um modulo do Node.js que utilizamos para enviar emails de confirmação para quando o cliente se regista no site, quando o cliente se esquece da *password* e a altera e também quando é registado trabalhadores e administradores. Começou a ser utilizado em 2010, quando não existia nenhuma boa opção para enviar mensagens de email, agora é a solução que mais se utiliza no Node.js.

As principais características do Nodemailer são:

- Tem um foco muito elevado em segurança.
- É um modulo único com zero dependências.
- Tem suporte par todo o tipo de caracteres incluindo emojis.
- Tal como qualquer outro modulo pode ser instalado com o npm num terminal do windows.
- Tem emails teste auto gerados pelo Etheral.email.
- Emails seguros usando TLS/STARTTLS.[27]
- Podemos manipular a mensagem que é enviada e enviar anexos.
- Tem autenticação OAuth3

3.18. Connect-flash

Connect-flash é um modulo do Node.js que utilizamos para guardar mensagens numa área especial da sessão. As mensagens escritas no flash e são eliminadas depois de serem mostradas ao utilizador.

Normalmente isto acontece quando o utilizador está a ser redirecionado para outra página, que acontece no nosso site quando um dos utilizadores faz uma ação numa página de

login para se redirecionar para outra página mostra uma mensagem a dizer que a operação foi bem-sucedida ou que foi malsucedida como mostra nas Figuras 3 e 4. [28]

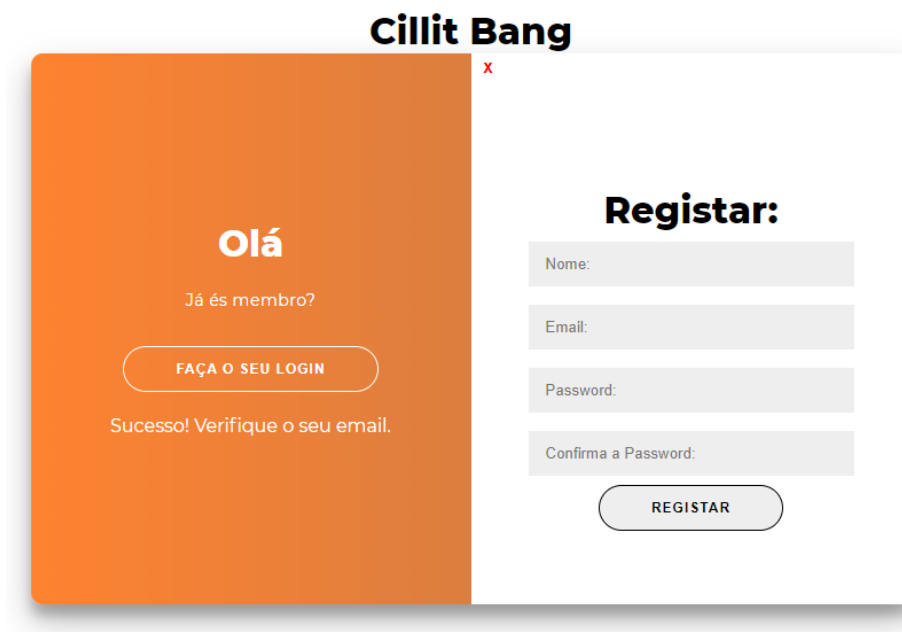


Figura 3 - Mensagem de Sucesso

Fonte: Printscreen tirado da página de login cliente do nosso site

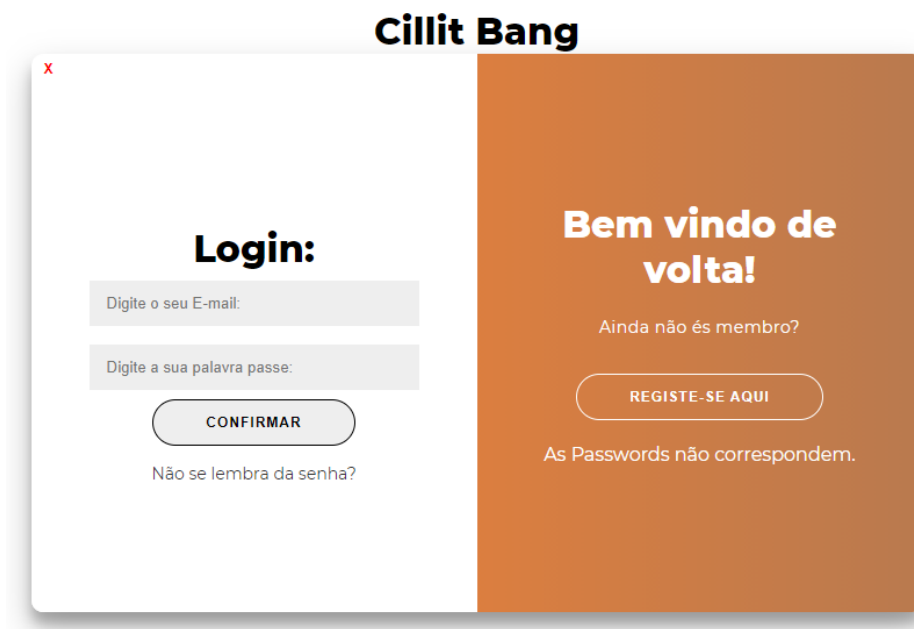


Figura 4 - Mensagem de insucesso

Fonte: Printscreen tirado da página de login do cliente do nosso site

3.19. Chart.js



Figura 5 – Total de trabalhadores

Fonte: Printscreens tirados da página dashboard do nosso site

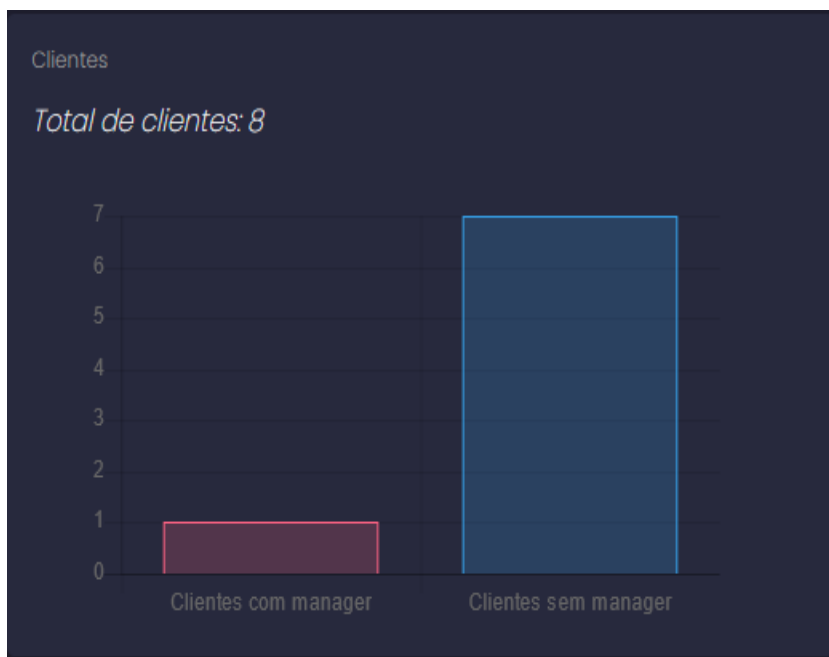


Figura 6 – Total de clientes

Fonte: Printscreens tirados da página dashboard do nosso site

Chart.js é uma biblioteca open-source JavaScript, utilizada para a visualização da informação sobre a empresa Cillit Bang como é possível ver nas Figuras 5 e 6. Foi criada pelo

desenvolvedor web Nick Downie em 2013, sendo que neste momento é mantida pela comunidade. Atualmente é a segunda ferramenta mais utilizada para visualização de dados sobre JavaScript no GitHub. É comumente caracterizada por ser bastante fácil na sua utilização, embora menos personalizável que outras ferramentas utilizadas no mercado. [29]

Com esta biblioteca é possível criar gráficos responsivos com qualquer nível de complexidade, com datasets esparsos e em tempo real com facilidade. Dispõe de 8 tipos diferentes de gráficos nomeadamente: de barras, de linha, de área, em tarte ou doughnut, em bolha, rada, polar e *scatter*. [30]

3.20. Canvas

Canvas é um elemento da HTML5 designado a delimitar uma área para processamento dinâmica dos gráficos [31].

4. Nome do primeiro capítulo (Desenvolvimento)

Neste capítulo, temos como objetivo explicar todo o processo da criação da nossa aplicação web, fazendo uma análise de página a página, referindo o seu propósito e as mais importantes features que as compõem.

Como ponto de partida, é importante referir como é que a nossa base de dados está organizada, de forma que seja mais fácil o entendimento da complexidade das relações entre dados existentes no nosso projeto.

Após termos apresentado como é que os nossos dados estão organizados, podemos dar início à análise das páginas destinadas aos clientes da nossa empresa, nomeadamente a página das boas-vindas (ou Welcome Page) e à página do cliente que acaba por se registar.

De seguida, realizamos uma análise da página destinada ao trabalhador, que efetua as respetivas limpezas marcadas pelos clientes.

Finalmente, analisamos a página de back office destinada aos administradores da empresa, onde será possível fazer a gestão dos recursos humanos da empresa, isto é, gerir clientes, trabalhadores, equipas de trabalhadores, marcações dos clientes, e de uma forma geral, acompanhar o crescimento da empresa.

Durante todo o processo de análise, iremos referir todas as funcionalidades oferecidas, bem como o raciocínio por detrás da construção das mesmas.

4.1. MongoDB Schemas

As bases de dados NoSQL, nomeadamente a que utilizámos MongoDB, utilizam schemas que, embora não têm exatamente o mesmo significado que os schemas SQL, partilham várias características em comum.

Um schema, no contexto do nosso projeto, é uma estrutura descrita na linguagem NoSQL e é suportada por um SGBD. O termo refere-se à organização da informação como uma planta, que explica como é que a base de dados está construída (dividida em coleções no caso de MongoDB ou em tabelas no caso de bases de dados relacionais). Citando uma definição mais formal: *"The formal definition of a database schema is a set of formulas (sentences) called integrity constraints imposed on a database."*

Os vários benefícios na escolha da base de dados MongoDB invés de uma base de dados relacional, acabaram por superar todas as desvantagens que nos deparámos ao longo do projeto. Por um lado, o MongoDB oferece uma grande facilidade na interação com os dados, visto que mapeia as estruturas de dados para aquelas mais utilizadas nas línguas de programação, o que leva a que os desenvolvedores tenham que escrever menos código e assim reduzirem o tempo da produção de aplicações. Outra grande vantagem é o facto de os schemas serem amigáveis à sua alteração, ou seja, não existe downtime associado à alteração dos schemas e não comprometem o bom funcionamento das operações.

As desvantagens mais sentidas na construção desta aplicação web utilizando NoSQL foram a falta de mecanismos para tornar mais fácil o relacionamento dos dados (das entidades) e a falta de funções de gestão da consistência dos dados. A primeira desvantagem não foi tão sentida como a segunda, visto que a nossa base de dados não depende de várias relações, tornando as relações existentes não muito complexas. A falta de consistência de dados manifestou-se pela falta de um algoritmo de garbage collection automático que fosse capaz de por exemplo, quando um documento é apagado, de apagar as suas respetivas referências nas entidades filhas.

Mesmo assim, tivemos uma boa experiência ao utilizar esta ferramenta e achamos que foi bem empregue para atingir o nosso objetivo final

4.1.1. User (Cliente)

```
const mongoose = require('mongoose');

//----- User Schema -----//
const UserSchema = new mongoose.Schema({
  name: {type: String,required: true},
  email: {type: String,required: true},
  password: {type: String,required: true},
  verified: {type: Boolean,default: false},
  role: { type: String, required: true, default: 'Cliente'},
  resetLink: {type: String,default: ''},
  admin: {type: mongoose.Schema.Types.ObjectId,ref: 'Admin'},
  hasAdmin: {type: Boolean,default: false},
  marcacaoCliente: [{type: mongoose.Schema.Types.ObjectId,ref: 'Marcacao'}],
  equipa: {type: String,default: 'Não'},
  feedbacks: [{type: mongoose.Schema.Types.ObjectId,ref: 'Feedback'}]
}, { timestamps: true });

const User = mongoose.model('User', UserSchema);

module.exports = User;
```

Figura 7 - Modelo usado para o cliente

Fonte: Printscreen tirado do modelo User do nosso código

O Schema User mostrado na Figura 7 representa um cliente da nossa empresa após ter-se registado na página Welcome.

Um cliente tem:

- Um nome;
- Um email;
- Uma password protegida com encriptação;
- Um papel ou um role, pode vir a ter um administrador associado;
- Um conjunto de marcações de limpezas;
- E um conjunto de feedbacks que possa vir a ter acerca do serviço fornecido.

O campo admin no schema do User representa uma relação de um para um a um administrador através do seu ID.

O campo marcacaoCliente é um array criado que contém as referências (os IDs) das limpezas agendadas que pertencem a um cliente em concreto. É representado por uma relação de um para muitos.

O campo feedbacks é um array criado para conter as referências (os IDs) dos feedbacks dados pelo próprio cliente. É representado por uma relação de um para muitos.

4.1.2. Trabalhador

```
const TrabSchema = new mongoose.Schema({
  name: {type: String,required: true},
  email: {type: String,required: true},
  password: {type: String,required: true},
  verified: {type: Boolean,default: false},
  role: { type: String, required: true, default: 'Trabalhador'},
  resetLink: {type: String,default: ''},
  pequena: {type: String,default: 'Não'},
  equipa: {type: Schema.Types.ObjectId,ref: 'Equipas'},
  marcTrab: [{type: Schema.Types.ObjectId,ref: 'Marcacao'}],
  note: {type: String,default: "Não tem nota"},
  disponibilidade: {type: String,default: "Sim"},
  feedbacksT: [{type: Schema.Types.ObjectId,ref: 'Feedback'}]
}, { timestamps: true });

const Trab = mongoose.model('Trab', TrabSchema);

module.exports = Trab;
```

Figura 8 - Modelo usado para o trabalhador

Fonte: Printscreen tirado do modelo Trab do nosso código

O Schema Trabalhador mostrado na Figura 8 representa um trabalhador da nossa empresa após ter sido registrado por um administrador.

Um trabalhador tem:

- Um nome;
- Um email;
- Uma password encriptada;
- Um role;
- Um campo que verifica se pertence a uma equipa;
- Um campo que referencia um objeto equipa;
- Um conjunto de ids de marcações que foram atribuídas à equipa em que o trabalhador pertence;
- Uma nota, isto é, uma pequena mensagem que pode ser deixada para outros administradores verem;
- Pode ter disponibilidade ou não para trabalhar;
- E tem um conjunto de ids de feedbacks que os próprios trabalhadores podem dar sob a forma de descontentamento, sugestão ou informação.

O campo `equipa` no Schema do Trabalhador representa uma relação de um para um entre uma equipa e um trabalhador, fazendo a referência de uma equipa em concreto através do seu `id`.

O campo `marcTrab` é um array criado que contém as referências (os IDs) das marcações que foram atribuídas à equipa em que o trabalhador pertence, que por sua vez foram agendadas pelos clientes. É representado por uma relação de um para muitos.

O campo `feedbacksT` é um array criado que contém as referências (os IDs) dos feedbacks que foram dados pelos próprios trabalhadores.

4.1.3. Administrador

```
const AdminSchema = new mongoose.Schema({
  firstName: {type: String, required: true},
  lastName: {type: String, required: true},
  email: {type: String, required: true},
  password: {type: String, required: true},
  verified: {type: Boolean, default: false},
  role: { type: String, required: true, default: 'Adminstrador'},
  resetLink: {type: String, default: ''},
  clients: [{type: mongoose.Schema.Types.ObjectId, ref: 'User'}]
}, { timestamps: true });

const Admin = mongoose.model('Admin', AdminSchema);

module.exports = Admin;
```

Figura 9 - Modelo usado para o Administrador

Fonte: Printscreen tirado do modelo Admin do nosso código

O Schema Administrador mostrado na Figura 9 representa um manager da nossa empresa após ter sido registado por outro administrador na página de back-office.

Um administrador tem:

- Um primeiro nome;
- Um último nome;
- Um email;
- Uma password encriptada;
- Um role;
- E um conjunto de clientes que gere.

O campo clients é um array criado que contém as referências (os IDs) dos clientes que o manager faz a gestão, isto é, o conjunto de clientes que estão associados a um manager em concreto. É representado por uma relação de um para muitos.

4.1.4. Marcação (Marcação da Limpeza)

```
const MarcacaoSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, required: true },
  date: { type: Date, required: true },
  hour: { type: String, required: true },
  address: { type: String, required: true },
  state: { type: String, default: 'Pendente' },
  equipa: { type: String, default: 'Não' },
  cliente: { type: Schema.Types.ObjectId, ref: 'User' },
  aval_admin: { type: String, default: "Sem Avaliação pelo Manager" },
  descricao: { type: String, default: "Sem Descrição" },
  aval_client: { type: String, default: "Sem Avaliação do Cliente" },
  avaliado: { type: Boolean, default: false },
  team: { type: Schema.Types.ObjectId, ref: 'Equipas' },
  avaliado_admin: { type: String, default: 'Não' },
},
{collection: 'Marcacao'})

module.exports = mongoose.model('Marcacao', MarcacaoSchema);
```

Figura 10 - Modelo usado para a Marcação

Fonte: Printscreen tirado do modelo Marcacao do nosso código

O Schema Marcação mostrado na Figura 10 representa uma marcação de uma limpeza agendada pelo cliente na Client Page.

Uma marcação tem:

- Um nome;
- Um email;
- Uma data;
- Uma hora;
- Uma morada;
- Um estado – Pendente ou Terminado;
- Um campo equipa que verifica se a marcação tem uma equipa associada;
- Uma referência ao cliente que a criou;
- Um campo aval_admin que verifica qual é a avaliação dada pelo administrador após o término da limpeza;

- Uma descrição dada pelos administradores à marcação;
- Um campo `aval_client` que verifica se a marcação foi avaliada por um cliente após o seu término;
- Um campo booleano `avaliado` que verifica se a marcação foi avaliada;
- Um campo `team` que faz uma referência a uma equipa que esteja associada com esta marcação;
- Um campo `avaliado_admin` que verifica se a marcação foi avaliada por um administrador após o seu término.

O campo `cliente` representa uma relação de uma marcação a um cliente, fazendo referência através do seu ID. Representa o cliente que agendou a limpeza.

O campo `team` é definido como uma relação de um para um, e é preenchido quando uma equipa é associada a uma marcação pelo administrador.

4.1.5. Equipas

```
const EquipaSchema = new mongoose.Schema({
  trab1: { type: Schema.Types.ObjectId, required: true, ref: 'Trab' },
  trab2: { type: Schema.Types.ObjectId, required: true, ref: 'Trab' },
  trab3: { type: Schema.Types.ObjectId, required: true, ref: 'Trab' },
  teamName: { type: String, required: true },
  marcsEquipa: [{ type: Schema.Types.ObjectId, ref: 'Marcacao' }],
},
{collection: 'Equipas'})

module.exports = mongoose.model('Equipas', EquipaSchema);
```

Figura 11- Modelo usado para a criação de equipas

Fonte: Printscreen tirado do modelo Equipas do nosso código

O Schema Equipas mostrado na Figura 11 representa uma única equipa de trabalhadores da nossa empresa após ter sido criada na página `/criar_equipa`. Uma equipa é composta por três trabalhadores.

Uma equipa tem:

- Um primeiro trabalhador;
- Um segundo trabalhador;
- Um terceiro trabalhador;
- Um nome de equipa;

- E um conjunto de marcações que foram associadas à equipa.

O campo `trab1`, `trab2` e `trab3` no Schema de Administrador representam três trabalhadores, referenciados pelos seus ids quando estes são selecionados na criação de uma equipa. Cada campo mencionado representa uma relação de um para um.

O campo `marcsEquipa` é um array criado que contém as referências (os IDs) das marcações que estão atribuídas a esta equipa. Quando uma equipa é associada a uma marcação de limpeza, os trabalhadores hereditam(?) essa mesma marcação. É representado por uma relação de muitos para um.

4.1.6. Feedback

```
const FeedbackSchema = new mongoose.Schema({
  name: {type: String, required: true},
  feedback: {type: String, required: true},
  cliente: {type: Schema.Types.ObjectId, ref: 'User'},
  trab: [{type: Schema.Types.ObjectId, ref: 'Trab'}]
}, { collection: 'Feedback' });

module.exports = mongoose.model('Feedback', FeedbackSchema);
```

Figura 12- Modelo usado para o Feedback

Fonte: Printscreen tirado do modelo Feedback do nosso código

O Schema Feedback mostrado na Figura 12 representa um feedback dado, tanto pelo cliente ou pelo trabalhador.

Um feedback tem:

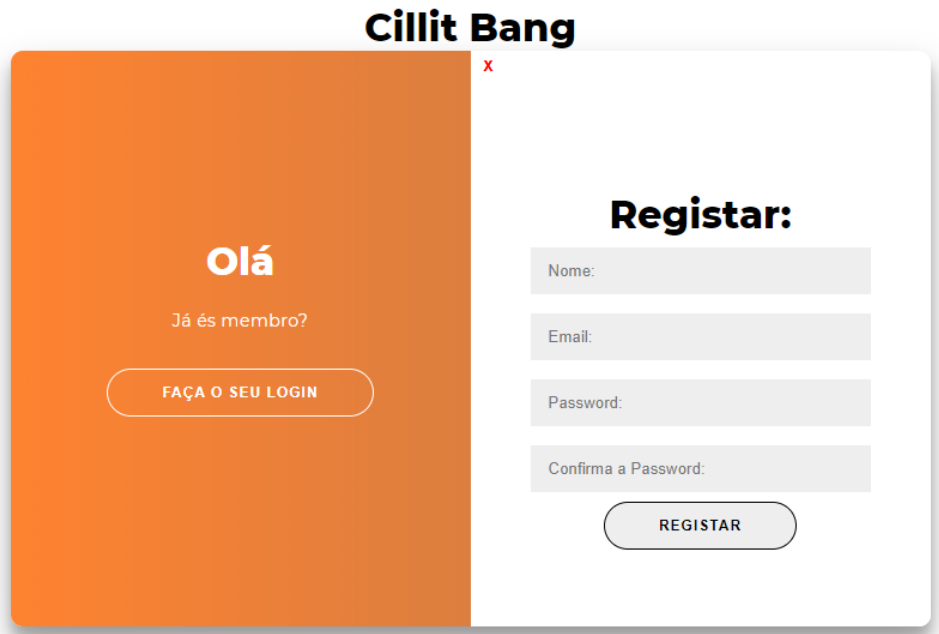
- Um nome do trabalhador ou cliente que enviou o feedback;
- O conteúdo do feedback;
- Uma referência a um cliente caso tenha sido o cliente a enviar;
- E uma referência a um trabalhador caso tenha sido um trabalhador a enviar.

O campo `cliente` representa uma relação de um feedback de um cliente caso tenha sido este a enviar na client page.

O campo `trabalhador` representa uma relação de um feedback de um trabalhador caso tenha sido este a enviar na worker page.

4.2. Funcionalidades das páginas

4.2.1. Login e Registo do cliente



Cillit Bang

Olá

Já és membro?

FAÇA O SEU LOGIN

Registrar:

Nome:

Email:

Password:

Confirma a Password:

REGISTAR

Figura 13- Formulário de registo do cliente

Fonte: Printscreen tirado da página de registar do cliente welcome do nosso site

Esta página de HTML é destinada aos clientes que pretendem entrar no site para se registar ou fazer login nas nossas empresas de limpezas. O cliente para se registar no site tem de clicar no botão “Registe-se Aqui” mostrado na Figura 13 onde vai aparecer o formulário de registo com os seguintes campos:

- Nome do cliente;
- Email do cliente (Tem que ter um formato de email senão não aceita);
- Password (Tem que ter um número, uma letra minúscula, uma letra maiúscula e no mínimo 8 caracteres);
- Confirmar a Password (Tem que ser exatamente igual a password escrita anteriormente);

Depois de clicar no botão de registar com todos os campos preenchidos vai ser executado um pedido POST e comparado com outros email já registados usando “User.findOne” se já existir uma conta com esse email vai aparecer uma mensagem a dizer que o email já foi registado e vai ter que repetir o processo, tendo os campos todos corretos o modulo JSON Web Token vai criar um token assinando o nome, email e password do cliente com a

chave privada para depois ser possível verificar que o token é legítimo, depois vai ser criado um link único com o token e com a ajuda no módulo nodemailer e é enviado para o email do cliente automaticamente para confirmar a sua conta. Ao clicar no link do email o módulo bcrypt vai transformar a password, ou seja, fazer um hash na password, quando isso acontece é praticamente impossível que esse novo string volte á password certa, por consequência a conta do cliente vai ser autenticada, ao voltar a clicar no link o código compara com os outros emails já registados usando o “User.findOne” e vai aparecer a mensagem que o email já tem uma conta associada e para fazer o login. Desde que tenha corrido tudo bem com a operação pois o link tem o tempo limite de 30 minutos vai ter a sua conta autenticada.

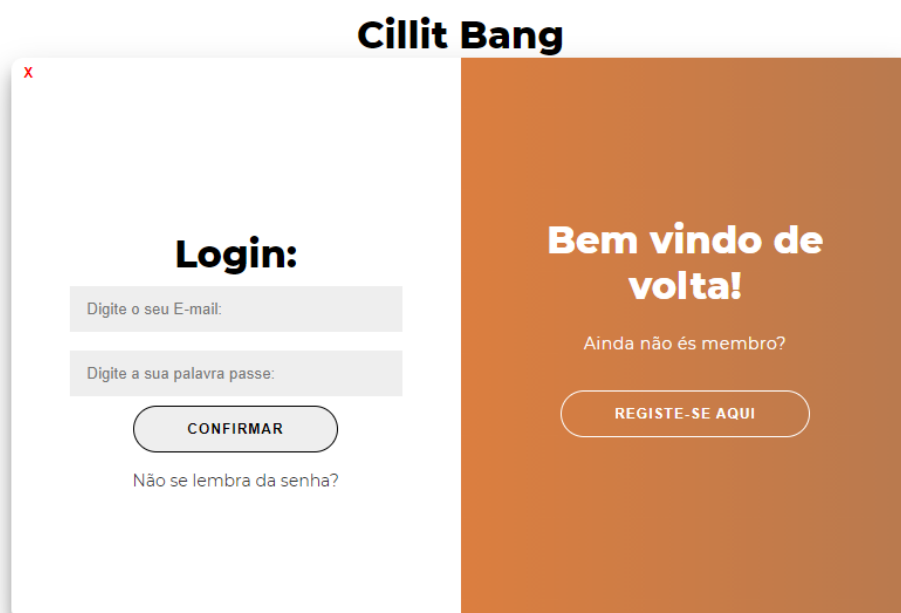


Figura 14- Campo de login do cliente

Fonte: Printscreen tirado da página de login do cliente do nosso site

Para fazer login é necessário preencher os campos com informações autenticadas do cliente, o seu email e a sua password. Depois de clicar no botão “Confirmar” mostrado na Figura 14 vai ser executado um pedido POST onde utilizamos o módulo passport com a estratégia do módulo passport-local para encontrar um email e id do cliente usando o “User.findOne” se o email estiver correto usamos o bcrypt para comparar a password se estes dois elementos estiverem verdadeiros o cliente é autenticado e serializado usando “passport.serializeUser” para gravar na sessão redirecionado para a página do cliente senão vai continuar na página de login e repetir o processo.

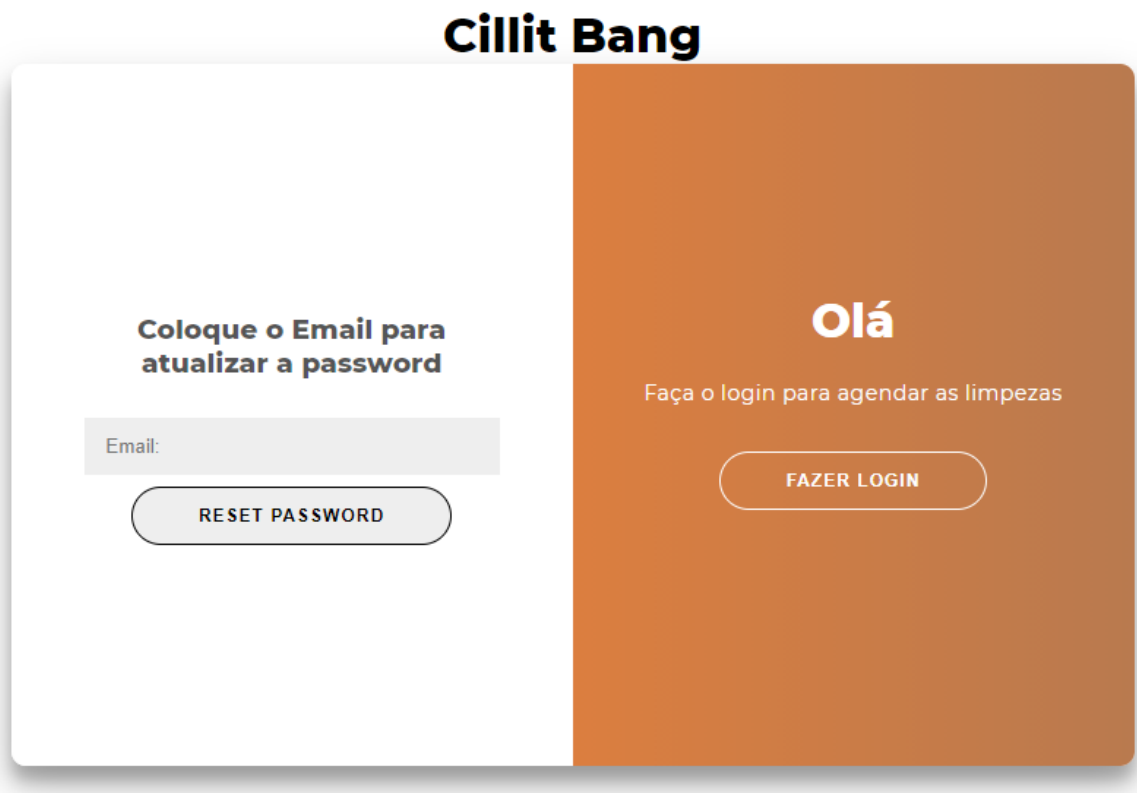


Figura 15 - Página de informação do email para repor a password

Fonte: Printscreen tirado da página “forgot” do cliente do nosso site

Se o cliente alguma vez esquecer da sua password pode clicar em “Não se lembra da senha?” como pode ver na Figura 14 em que vai ser redirecionado para a página “forgot” onde pode introduzir o seu email como pode observar na Figura 15, onde vai correr um pedido POST e a função vai tentar encontrar o seu email na base de dados usando novamente “User.findOne” se não encontrar vai ter que criar uma conta nova. Ao encontrar o email vai ser enviado um email usando o módulo nodemailer da mesma maneira que foi usado para enviar o email de confirmação de conta, mas agora com “User.updateOne” para atualizar o token e enviar outro link para o email do cliente.

The image shows a web interface divided into two vertical panels. The left panel has a white background and contains the heading "Coloque a nova password para a atualizar" in bold. Below this are two input fields: "Enter Password" and "Confirm Password", both with light gray borders. At the bottom of this panel is a rounded button labeled "ATUALIZAR PASSWORD". The right panel has an orange background and features the word "Olá" in large white font. Below it, the text "Faça o seu login para ver os seus filmes favoritos" is displayed. At the bottom of the right panel is a rounded button labeled "FAZER LOGIN".

Figura 16- Campos para repor a password

Fonte: Printscreen tirado da página de “reset” do cliente do nosso site

Ao clicar no link e se o link não expirar pois tem novamente um limite de 30 minutos, vai executar “User.findById” para encontrar o id certo do cliente e ser redirecionado para a página de “reset” como pode ver na Figura 16, colocando as passwords exatamente iguais e ao clicar no botão “Atualizar password”, vai ser feito um pedido de POST, o módulo bcrypt vai encriptar a password com hash, correndo a função “User.findByIdAndUpdate” procura pelo cliente usando o seu id correto e atualiza a password. Por fim vai ser redirecionado para a página de login podendo assim fazer login com a sua nova password.

4.2.2. Login e Registo do trabalhador

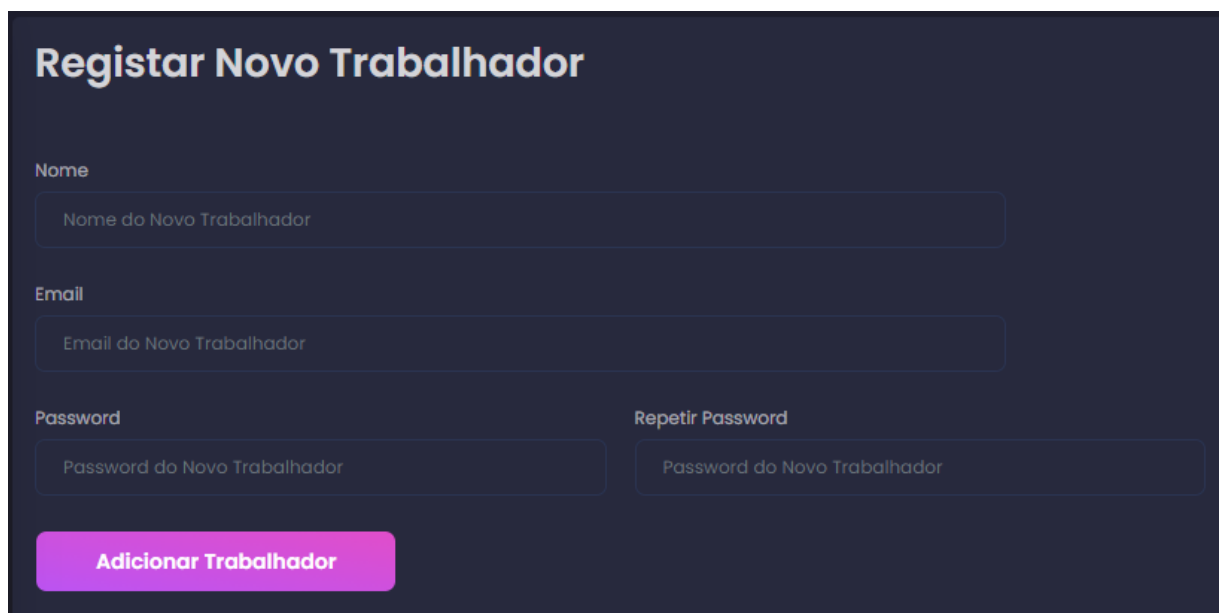


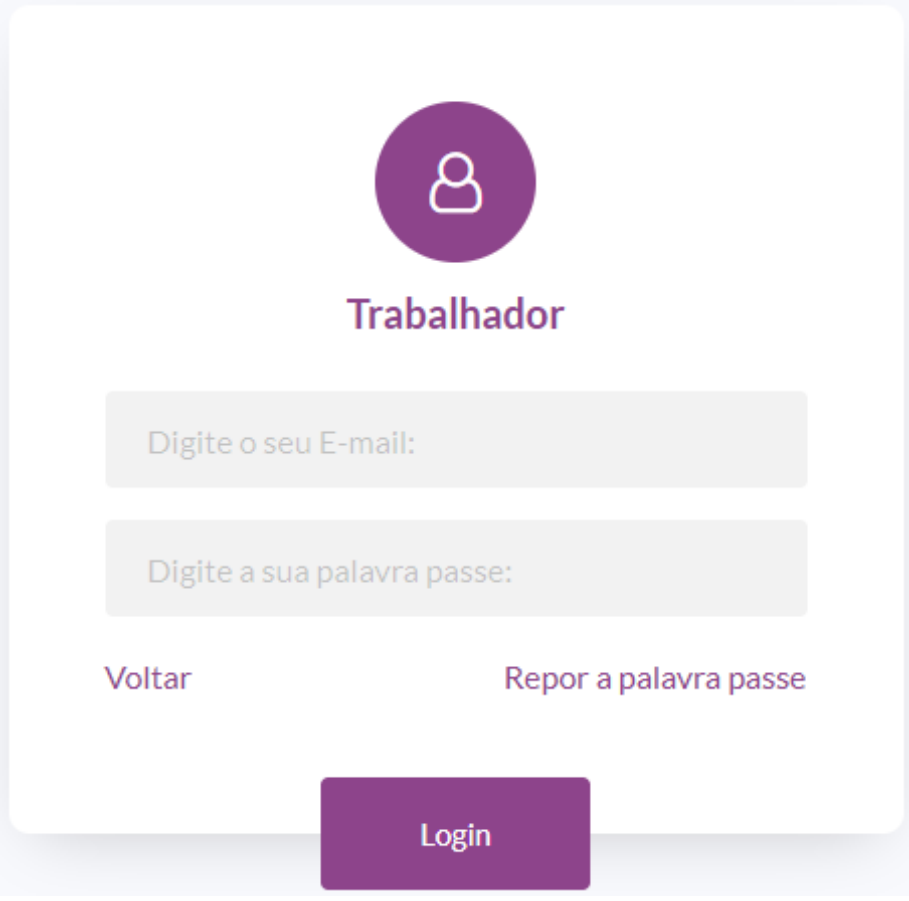
Figura 17- Formulário de registo de um novo trabalhador

Fonte: Printscreen tirado da página de registo de trabalhador do nosso site

Para registar um trabalhador é necessário que seja um administrador a fazê-lo na página de registo de staff depois de um administrador fazer login para a parte do site destinada a back office. O administrador vai ter que preencher o formulário de registo dos trabalhadores mostrado na Figura 17 com os seguintes campos:

- Nome do Trabalhador;
- Email dado pela empresa para o Trabalhador (tem que ter o formato de um email);
- Password (Tem que ter um número, uma letra minúscula, uma letra maiúscula e no mínimo 8 caracteres);
- Confirmar a Password (Tem que ser exatamente igual a password escrita anteriormente);

Depois de clicar no botão de “Adicionar Trabalhador” com todos os campos preenchidos vai ser executado um pedido POST e comparado com outros email já registados usando “Trab.findOne” se já existir uma conta com esse email vai aparecer uma mensagem a dizer que o email já foi registado e vai ter que repetir o processo, tendo os campos todos corretos o código vai fazer todos os passos exatamente iguais ao registo de clientes no ponto 4.2.1. Por fim o administrador vai clicar no link enviado para o email do trabalhador autenticando a conta.

A login form for an employee. At the top is a purple circle icon with a white person silhouette. Below it is the word "Trabalhador" in bold purple. There are two light gray input fields: the first is labeled "Digite o seu E-mail:" and the second is labeled "Digite a sua palavra passe:". Below the first field is a purple link "Voltar". Below the second field is a purple link "Repor a palavra passe". At the bottom is a purple button labeled "Login".

Trabalhador

Digite o seu E-mail:

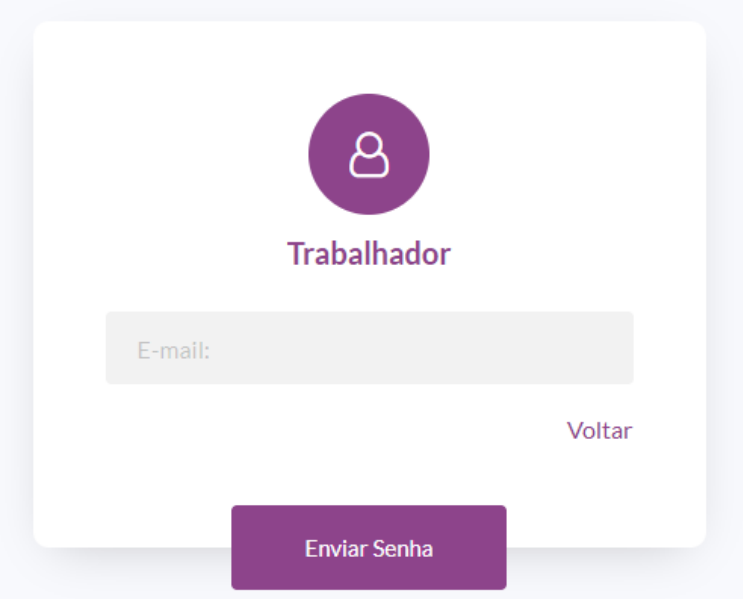
Digite a sua palavra passe:

[Voltar](#) [Repor a palavra passe](#)

Login

Figura 18- Campos de login do trabalhador

Fonte: Printscreen tirado da página do login to trabalhador do nosso site

A form to indicate an employee's email. At the top is a purple circle icon with a white person silhouette. Below it is the word "Trabalhador" in bold purple. There is a light gray input field labeled "E-mail:". To the right of the input field is a purple link "Voltar". At the bottom is a purple button labeled "Enviar Senha".

Trabalhador

E-mail:

[Voltar](#)

Enviar Senha

Figura 19- Campo para indicar o email do trabalhador

Fonte: Printscreen tirado da página de indicar o email do trabalhador do nosso site

The image shows a web form for a 'Funcionario' (Employee) to reset their password. At the top, there is a purple circular icon with a white person silhouette, followed by the title 'Funcionario' in bold purple text. Below this, there are two light gray input fields. The first field is labeled 'Coloque a nova password:' and the second is labeled 'Confirme a password:'. To the right of the second field is a purple link labeled 'Voltar'. At the bottom center, there is a large purple button with the text 'Atualizar Senha' in white.

Figura 20- Campos para repor a password do trabalhador

Fonte: Printscreen tirado da página de repor a password do trabalhador do nosso site

Neste momento o trabalhador vai poder fazer o login de maneira igual ao cliente descrito no ponto 4.2.1, mas vai ser redirecionado para outra página, a página do trabalhador.

Se o trabalhador se esquecer da sua password ou a quiser mudar pode o fazer clicando em “Repor a palavra-passe” em que o método também é igual e descrito no ponto 4.2.1, mas vão ser redirecionado para outras páginas, a página em que se coloca o email para encontrar a conta correta mostrada na Figura 19 e a página em que coloca a password nova mostrada na Figura 20

4.1.1. Login e Registo do administrador

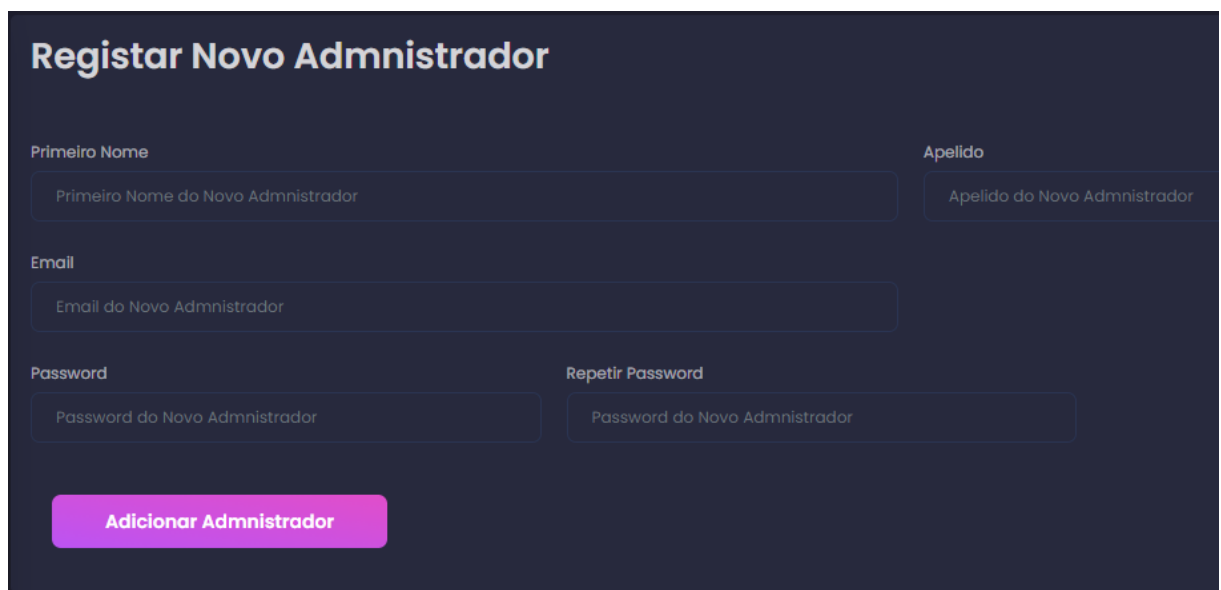
A imagem mostra um formulário web para registar um novo administrador. O formulário tem um fundo escuro e o título "Registar Novo Adminstrador" em branco. Os campos de entrada são: "Primeiro Nome" e "Apelido" (dois campos lado a lado no topo), "Email" (um campo centralizado abaixo dos anteriores), "Password" e "Repetir Password" (dois campos lado a lado na base superior). Cada campo tem um placeholder com o texto "Primeiro Nome do Novo Adminstrador", "Apelido do Novo Adminstrador", "Email do Novo Adminstrador", "Password do Novo Adminstrador" e "Password do Novo Adminstrador" respetivamente. Um botão laranja com o texto "Adicionar Adminstrador" está na base do formulário.

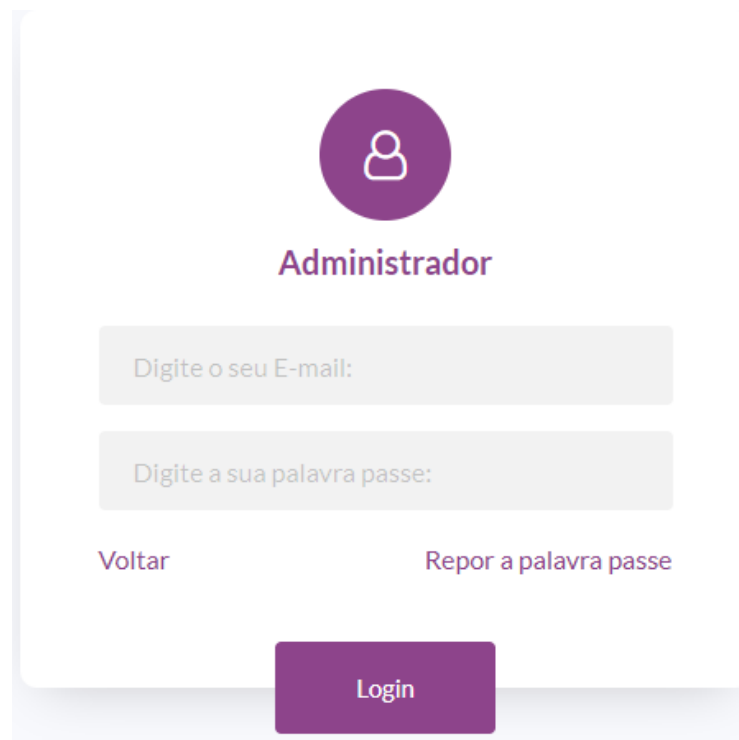
Figura 21- Formulário de registo de administradores

Fonte: Printscreen tirado da página de registo de administrador do nosso site

Para registar um administrador é necessário que seja outro administrador a fazê-lo na página de registo de administrador na parte do site destinada a back office. O administrador vai ter que preencher o formulário de registo com os seguintes campos:

- Nome do Administrador;
- Email dado pela empresa (tem que ter o formato de um email);
- Password (Tem que ter um número, uma letra minúscula, uma letra maiúscula e no mínimo 8 caracteres);
- Confirmar a Password (Tem que ser exatamente igual a password escrita anteriormente);

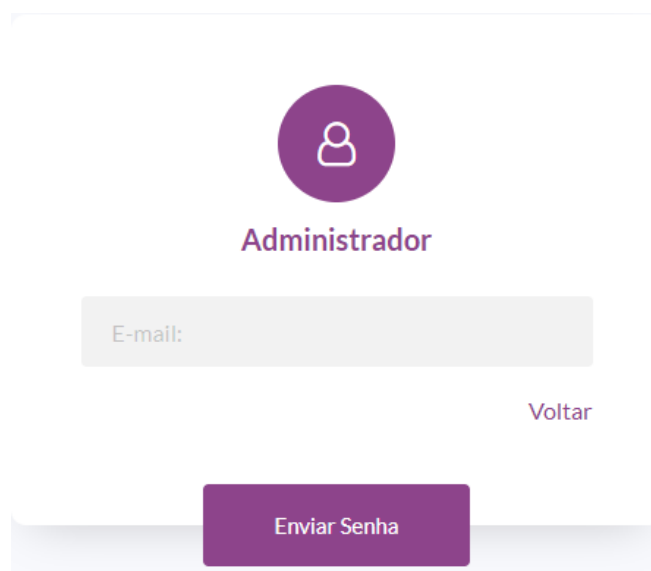
Depois de clicar no botão de “Adicionar Administrador” com todos os campos preenchidos vai ser executado um pedido POST e comparado com outros email já registados usando “Admin.findOne” se já existir uma conta com esse email vai aparecer uma mensagem a dizer que o email já foi registado e vai ter que repetir o processo, tendo os campos todos corretos o código vai fazer todos os passos exatamente iguais ao registo de clientes no ponto 4.2.1. Por fim o administrador vai clicar no link enviado para o email autenticando a conta.



The image shows a login form for an administrator. At the top, there is a purple circular icon with a white person silhouette. Below it, the word "Administrador" is written in bold purple text. There are two input fields: the first is labeled "Digite o seu E-mail:" and the second is labeled "Digite a sua palavra passe:". Below the password field, there are two links: "Voltar" and "Repor a palavra passe". At the bottom, there is a purple button labeled "Login".

Figura 22- Campo de login do administrador

Fonte: Printscreen tirado da página do login to trabalhador do nosso site



The image shows a form for indicating the administrator's email. At the top, there is a purple circular icon with a white person silhouette. Below it, the word "Administrador" is written in bold purple text. There is one input field labeled "E-mail:". To the right of the input field, there is a link labeled "Voltar". At the bottom, there is a purple button labeled "Enviar Senha".

Figura 23- Campo para indicar o email do administrador

Fonte: Printscreen tirado da página de indicar o email do administrador do nosso site

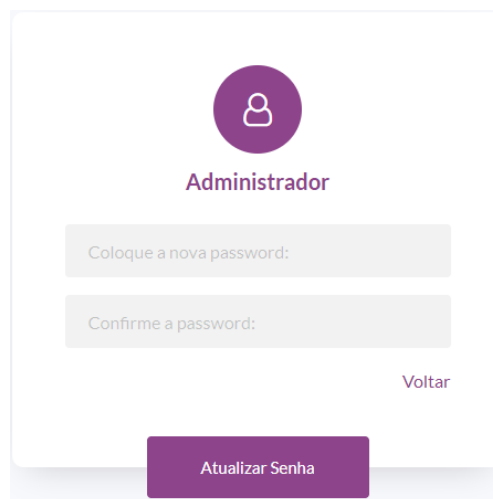
A web form for an administrator to reset their password. At the top, there is a purple circular icon with a white person silhouette, followed by the text "Administrador" in purple. Below this are two light gray input fields: the first is labeled "Coloque a nova password:" and the second is labeled "Confirme a password:". To the right of the second field is a purple link labeled "Voltar". At the bottom center is a large purple button labeled "Atualizar Senha".

Figura 24- Campos para repor a password do administrador

Fonte: Printscreen tirado da página de indicar o email do administrador e para repor a password do administrador do nosso site

Neste momento novo administrador vai poder fazer o login de maneira igual ao cliente descrito no ponto 4.2.1, mas vai ser redirecionado para outra página, a página do de back office só destinada a administradores.

Se o novo administrador se esquecer da sua password ou a quiser mudar pode o fazer clicando em “Repor a palavra-passe” em que o método também é igual e descrito no ponto 4.2.1, mas vão ser redirecionado para outras páginas, a página em que se coloca o email para encontrar a conta correta mostrada na Figura 23 e a página em que coloca a password nova mostrada na Figura 24.

4.1.2. Welcome Page



Figura 25- Imagem da página welcome

Fonte: Printscreen tirado da página welcome do nosso site

Esta página HTML mostrada na figura 25 é destinada ao cliente que ainda não se registou ou que ainda não tenha feito o login, e tem como principal objetivo apresentar ao potencial cliente, os serviços fornecidos pela nossa empresa de limpezas. Para além disso, tem como função de ser a página por onde os trabalhadores e administradores possam fazer o login para as suas respetivas páginas.

Numa primeira análise o cliente é deparado por um carousel criado com Bootstrap, com várias imagens seguidas por uma pequena descrição, indicativas da finalidade dos nossos serviços.

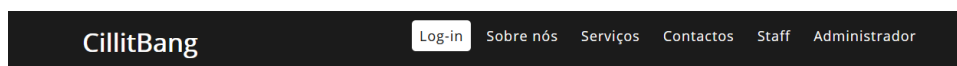


Figura 26- Menu de navegação da página welcome

Fonte: Printscreen tirado da barra de navegação da página welcome do nosso site

No menu de navegação mostrado na Figura 26 o cliente poderá fazer o login, que por sua vez será redirecionado para uma página onde pode introduzir as suas credenciais no caso de já estar registado, ou de preencher o formulário para poder se registar. Ainda no menu, o cliente pode clicar no “Sobre nós” e será redirecionado para uma secção onde poderá conhecer mais sobre a nossa empresa e o nosso modus operandi. Na área “Serviços”, o cliente fica a conhecer os nossos serviços e todas indicações necessárias para proceder a uma marcação de limpeza.

4.1.3. Client Page



Figura 27- Imagem da página de cliente

Fonte: Printscreen tirado da página cliente do nosso site

Esta página mostrada na Figura 27 destina-se exclusivamente ao cliente que já se registou e que já fez o login. Modelada de forma semelhante com Bootstrap, esta página permite ao utilizador fazer uma marcação de uma limpeza, verificar as suas marcações agendadas e ainda fornecer feedback aos managers.

Área do Cliente

Marcação da Limpeza Verificar Marcações Feedback sobre as suas Limpezas

Marque a sua limpeza

Primeiro e Último nome

Email

Escolha uma data:

Escolha uma hora:

dd/mm/aaaa

--:--

Morada da marcação:

Morada

Marcar

Figura 28- Secção de marcação da página do cliente

Fonte: Printscreen tirado da secção “Área do Cliente” da página cliente do nosso site

A “Área do Cliente” mostrada na Figura 28 é uma secção com um sub-menu que permite ao cliente utilizar as funcionalidades acima referidas:

- Agendar uma limpeza. O cliente preenche o formulário com o seu nome, o email, a data de agendamento, a respetiva hora e a morada da marcação. Assim que o cliente submeta o formulário, é enviado um pedido de POST com a nova marcação a ser introduzida na base de dados. Quando o pedido tiver sido bem-sucedido é automaticamente atribuído à marcação os campos: equipa, aval_admin, descrição, aval_client, state e avaliado_admin, todos com valores default. Para além disso é estabelecida a relação de cliente e marcação, materializado por uma referência do id do cliente na marcação.
- Verificar as marcações. O cliente terá uma tabela com todas as suas marcações agendadas. Na tabela, as marcações são identificadas e fornecem informação acerca do seu estado – Pendente ou Terminado. No caso de uma limpeza estiver terminada, o cliente poderá avaliá-la de 1 a 5 na coluna “Avaliar”. Este dropdown selector aparecerá apenas se a limpeza tiver sido terminada e que ainda não tenha sido avaliada. Para mostrar os conteúdos da tabela foi necessário realizar um pedido GET com base no id do cliente que está neste momento a verificar as suas marcações. A query para a base de dados foi realizada com base no id do utilizador na sessão fornecido pela ferramenta passport.js. Ao encontrar o utilizador em questão, procede-se à utilização de uma função mongoose denominada populate. Esta função tem como objetivo aceder ao array das marcações do cliente e devolver em formato JSON, todos os objetos marcações

do cliente. A tabela é construída assincronamente e dinamicamente, através de uma função Ajax que preenche a tabela HTML em tempo real.

- Fornecer feedback aos managers da empresa. O cliente terá que preencher um formulário com o seu nome e o seu feedback que, quando este é submetido, será guardado na coleção Feedback como um objeto que referencia o feedback de um cliente.

4.1.4. Worker Page



Figura 29- Barra de navegação do trabalhador

Fonte: Printscreen tirado da barra de navegação da página trabalhador do nosso site

Esta página destina-se exclusivamente ao trabalhador que já foi registado por um administrador, e que já fez o login. Sendo bastante semelhante à página do cliente como é mostrado na Figura 29, esta permite ao trabalhador verificar as suas limpezas e fornecer feedbacks aos administradores.

Área do Trabalhador

Nome do Trabalhador:

Verificar Marcações

Feedback do Trabalhador

Nome do Trabalhador

Morada	Data	Hora	Estado	Terminar	Avaliação
12	1111-11-11T00:00:00.000Z	11:11	Pendente	<div>Terminar</div>	
sdqc2	1111-11-11T00:00:00.000Z	21:32	Pendente	<div>Terminar</div>	
lodej13045@godpeed.com	2021-07-19T00:00:00.000Z	12:12	Pendente	<div>Terminar</div>	

Figura 30- Área do trabalhador

Fonte: Printscreen tirado da secção “Área do Trabalhador” da página trabalhador do nosso site

A “Área do Trabalhador” mostrada na Figura 30 é uma secção que dispõe de um submenu que permite ao trabalhador utilizar as funcionalidades acima referidas:

- Verificar as limpezas. O trabalhador poderá verificar as suas limpezas do mesmo modo que o cliente o faz. As únicas diferenças prendem-se em duas colunas da tabela. A coluna “Terminar” tem como objetivo apresentar um botão “Terminar”, que quando clicado, vai alterar o estado da limpeza de Pendente para Terminado. Esta alteração possibilita ao cliente e ao administrador, avaliarem o trabalho realizado. Quando terminado, o botão é removido. Quando avaliado pelo administrador, a coluna avaliação apresentará a nota que o administrador deu. A construção da tabela seguiu os mesmos moldes que a construção da tabela de verificação das marcações do cliente. Foi necessário fazer uma query com base no id do utilizador em sessão, e utilizada a função populate da biblioteca mongoose para devolver todas as marcações do array das marcações do trabalhador. Este array, marcTrab, vai herdar as marcações que foram atribuídas à equipa a que o trabalhador em questão pertence, ou seja, quando uma equipa é a associada a uma limpeza, todos os seus trabalhadores vão herdar essa limpeza.
- Fornecer feedback. Tal como o cliente, o trabalhador poderá fornecer feedback aos administradores da empresa. Terá que preencher o formulário com o seu nome e o feedback. Quando este for submetido utilizando um pedido POST, será guardado na coleção Feedback como um objeto que referencia o feedback do trabalhador.

4.1.5. Página de Back-office (Página dos administradores)

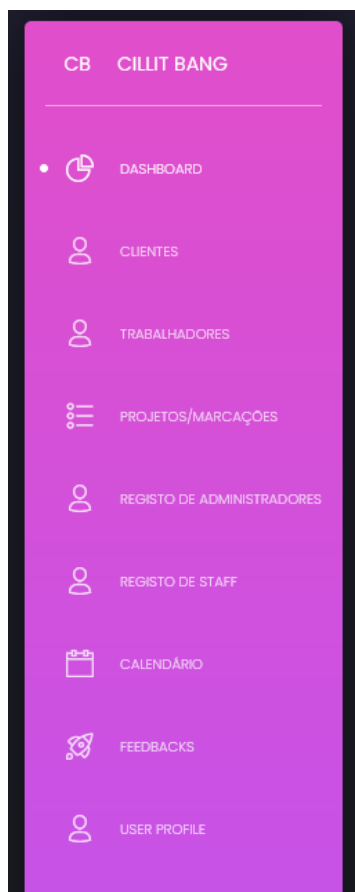


Figura 31- Barra de navegação do administrador

Fonte: Printscreen tirado da barra de navegação da página administrador do nosso site

Esta página destina-se exclusivamente a administradores, após terem sido registados por outro administrador e terem feito o login. O back-office tem como principal objetivo fornecer um ambiente de gestão dos recursos humanos da empresa aos administradores. Na Figura 31, verifica-se um menu com todos os conteúdos necessários para atingir o objetivo final da aplicação.

Os conteúdos são:

- Dashboard: é apresentado ao administrador uma página que fornece uma visão geral do estado da empresa e do negócio. É composta pela reunião de gráficos informativos criados com chart.js;
- Clientes: uma área em que o administrador tem acesso a todas as informações dos clientes da empresa;
- Trabalhadores: uma área em que o administrador tem acesso a todas as informações dos trabalhadores da empresa;

- **Projetos/Marcações:** uma área em que o administrador tem acesso a todas as informações relativas às marcações de limpeza dos clientes;
- **Registo de Administradores:** para o registo de novos administradores;
- **Registo de Staff:** para o registo de novos trabalhadores;
- **Calendário:** um calendário partilhado por todos os administradores para estes poderem organizar todo o tipo de tarefas relativas à gestão dos recursos humanos;
- **Feedbacks:** uma área em que é possível verificar todos os feedbacks enviados seja por clientes ou trabalhadores;
- **User profile:** uma área que reúne todas as informações relativas ao utilizador que está neste momento a utilizar a aplicação.

4.1.6. Dashboard

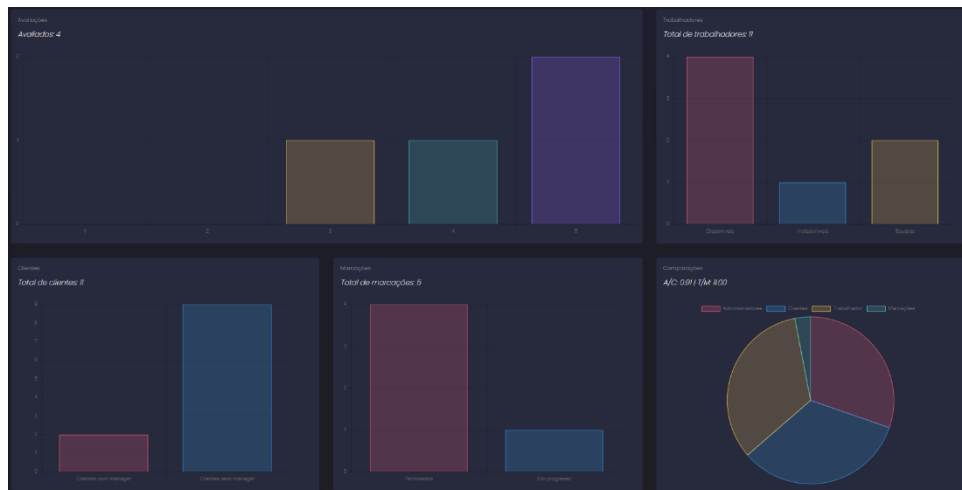


Figura 32- Imagem do dashboard

Fonte: Printscreen tirado da barra de navegação da página administrador do nosso site

A figura 32 mostra a página do dashboard que nos fornece uma visão geral do estado da empresa e do negócio, é composta por um conjunto de cinco gráficos que foram criados com ajuda do Chart.js, canvas, e funções assíncronas, `async`. A função torna-se assíncrona se for marcada com `async` antes, `async function`. É necessário utilizar a função assíncrona pois vamos utilizar o `fetch()`, `const response = await fetch('URL')`, que inicia uma solicitação e retorna um promise, quando a solicitação é concluída, o promise é transformado em um response object que é um espaço genérico para vários formatos de dados, devido ao fato de termos de esperar a

conclusão da solicitação utilizamos o `await` que pausa a função até que a solicitação seja concluída. De seguida utilizamos `const data = await response.json()` que extrai os dados json object da response e associa os a constante data, transformando a num array dos dados [32][33].

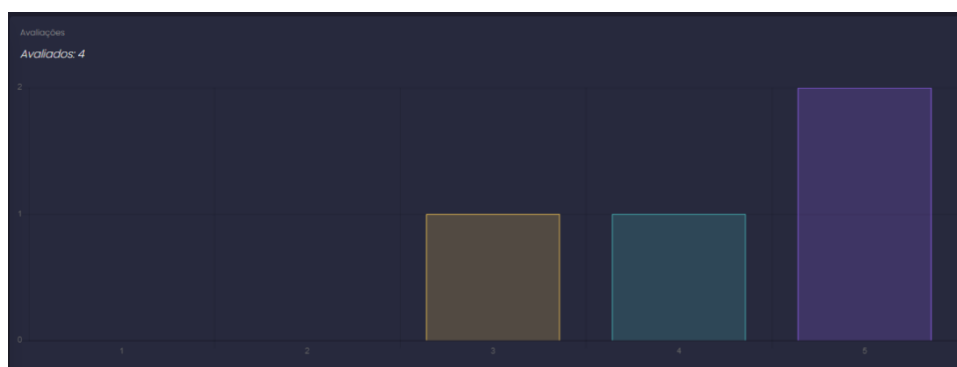


Figura 33- Gráfico de avaliações

Fonte: Printscreen tirado do dashboard da página administrador do nosso site

A figura 33 demonstra-nos o gráfico das avaliações deixadas pelos clientes que por sua vez demonstra-nos a qualidade do serviço fornecido pela empresa, pois um dos fatores mais importantes com o qual a empresa deve preocupar-se é a satisfação do cliente.

Para criar este gráfico utilizamos uma função assíncrona com sete arrays cinco deles são utilizados para guardar as devidas avaliações, ou seja, um array guarda as avaliações de nota 1 o segundo array guarda as avaliações de nota 2 e assim por diante, e o sétimo array guarda as marcações sem avaliações, para preencher os arrays utilizamos um for loop, que começa no zero e é limitado pelo total de marcações existentes, e uma condição if e cinco condições else if onde a primeira condição verifica se marcação tem ou não avaliação caso a marcação não tenha avaliação ela é guardada no sétimo array, as restantes cinco condições verificam a nota, de 1 a 5, atribuída a marcação, após preenchimento dos cinco arrays com as devidas notas o sexto array é preenchido com o tamanho dos cinco arrays ordenadamente, o primeiro tamanho é do array das avaliações com nota 1 o segundo tamanho é do array com a nota 2 etc., o sétimo array permite calcular o total das marcações avaliadas fazendo subtração do total das marcações com o tamanho do sétimo array. A segunda função assíncrona é a função que desenha o gráfico, mas antes de desenhá-lo ela chama a primeira função, de recolha de dados, e aguarda a obtenção do retorno da primeira função para prosseguir com preenchimento dos gráficos.

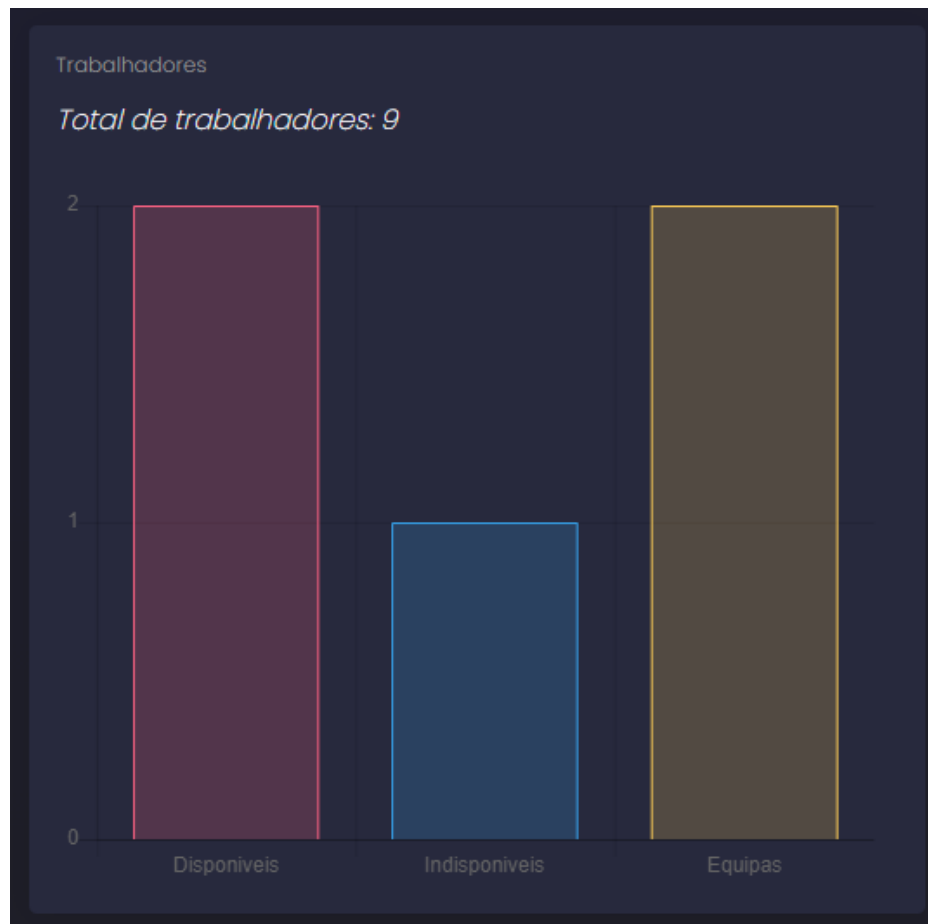


Figura 34- Gráfico total dos trabalhadores

Fonte: Printscreen tirado do dashboard da página administrador do nosso site

O gráfico da figura 34 demonstra-nos o total dos trabalhadores como também quantos trabalhadores de momento estão disponíveis, indisponíveis e quantas equipas de três trabalhadores existem.

Para criar o gráfico utilizamos três funções assíncronas a primeira é utilizada para obter as informações de trabalhadores da base de dados para isso utilizamos dois arrays o primeiro para guardar todos os todos os trabalhadores e o segundo para guardar os trabalhadores indisponíveis para isso utilizamos um for loop, que começa no zero e é limitado pelo número dos trabalhadores existentes na base de dados, e duas condições if a primeira verifica se o trabalhador pertence a equipa, caso ele não pertença ele é guardado no primeiro array e a segunda condição if verifica a disponibilidade do trabalhador caso ele esteja indisponível ele é guardado no segundo array. O tamanho destes dois arrays permite-nos calcular os trabalhadores disponíveis fazendo a subtração do tamanho do array dos trabalhadores com o tamanho do array dos trabalhadores indisponíveis e também o tamanho do array dos trabalhadores indisponíveis indica-nos o número deles. A segunda função assíncrona cria um array a partir da base de dados

com equipas existentes o tamanho deste array indica-nos o número delas. A terceira função assíncrona faz chamada a primeiras duas, após o que espera o retorno dos dados destas funções para prosseguir com a criação dos gráficos.

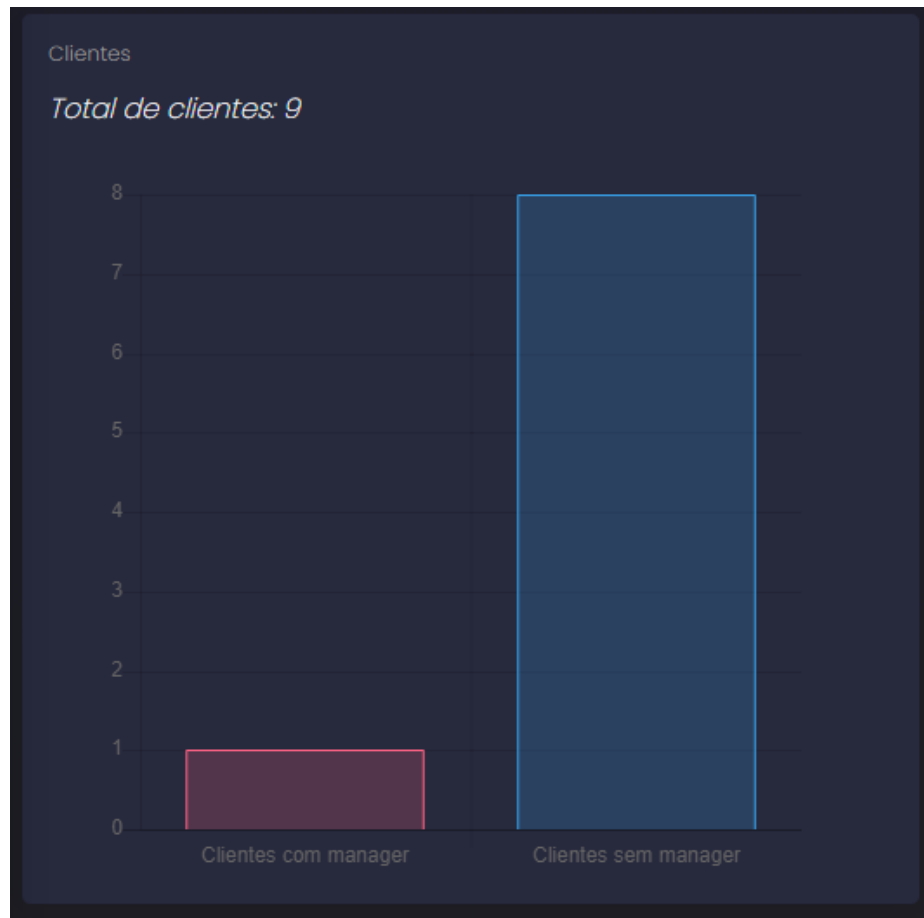


Figura 35- Gráfico do total de clientes

Fonte: Printscreen tirado do dashboard da página administrador do nosso site

O gráfico da figura 35 demonstra-nos o total dos clientes, quantos clientes já possuem um manager como também aos quantos falta associar um.

Para criar o gráfico utilizamos duas funções assíncronas a primeira é utilizada para obter as informações dos clientes da base de dados e guarda-las em dois arrays, o primeiro array é para guardar os clientes com manager associado e o segundo é para guardar os clientes sem um manager associado, para preencher arrays utilizamos um for loop, que começa no zero e é limitado pelo número de clientes existentes na base de dados, com uma condição if que verifica se o cliente tem um manager associado caso tenha manager associado é guardado no primeiro array, se não tiver e utilizada uma condição else para guarda lo no segundo array, também utilizamos o terceiro array para guardar os tamanhos dos arrays anteriores, o total dos clientes

é indicado pelo número dos clientes existentes na base de dados. A segunda função chama a primeira e espera até ela retornar os dados após o que ela prossegue para a criação dos gráficos.

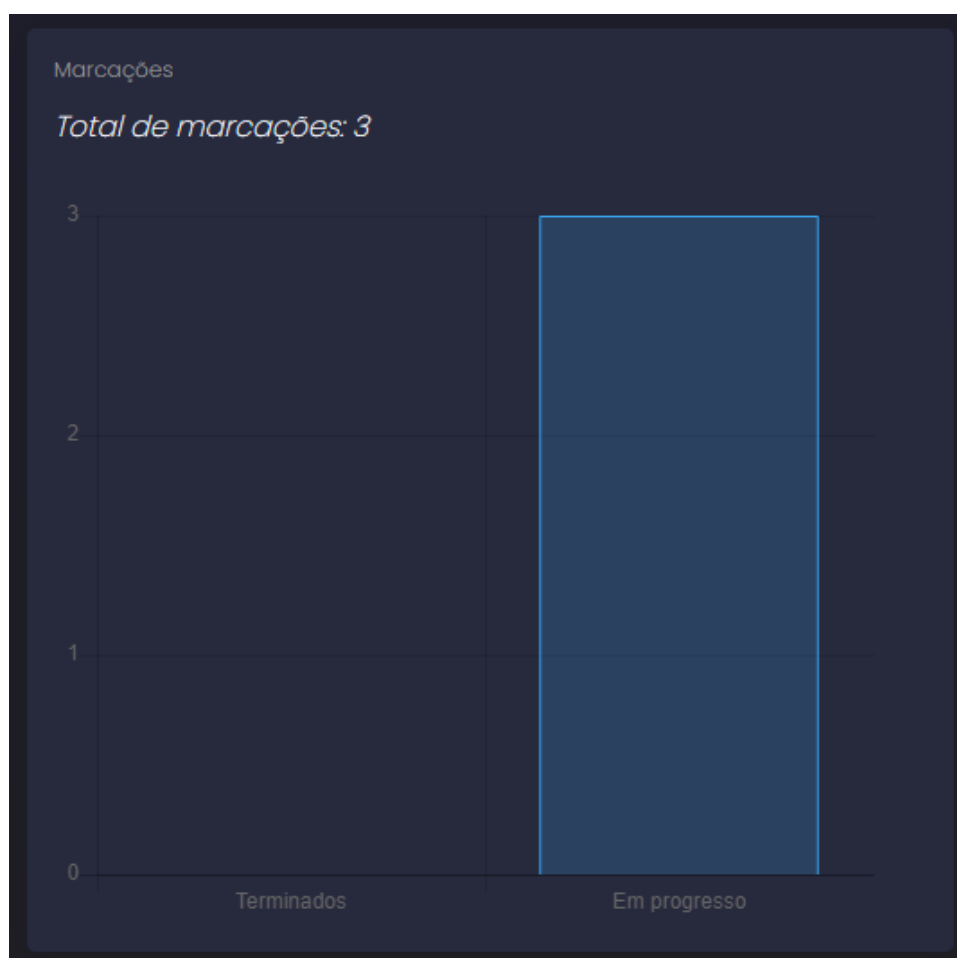


Figura 36- Total de marcações

Fonte: Printscreen tirado do dashboard da página administrador do nosso site

A figura 36 representa o gráfico das marcações, onde se pode observar o total delas as que já foram terminadas e as que ainda estão em processo.

Para criar o gráfico utilizamos duas funções assíncronas a primeira é utilizada para obter informação das marcações da base de dados e guarda as num array de seguida é utilizado um for loop, que inicia no zero e é limitado pelo tamanho do array das marcações, e uma condição if para verificar se a marcação tem uma avaliação ou não, se a marcação tiver uma avaliação ela é guardada num novo array, array das marcações sem avaliação, o tamanho do array das marcações sem avaliação indica nos quantas marcações estão em progresso e a subtração do array das marcações com array sem avaliações indica-nos o número das marcações dadas por

terminado. A segunda função faz chamada a primeira após o que aguarda pela sua resposta para criar os gráficos.

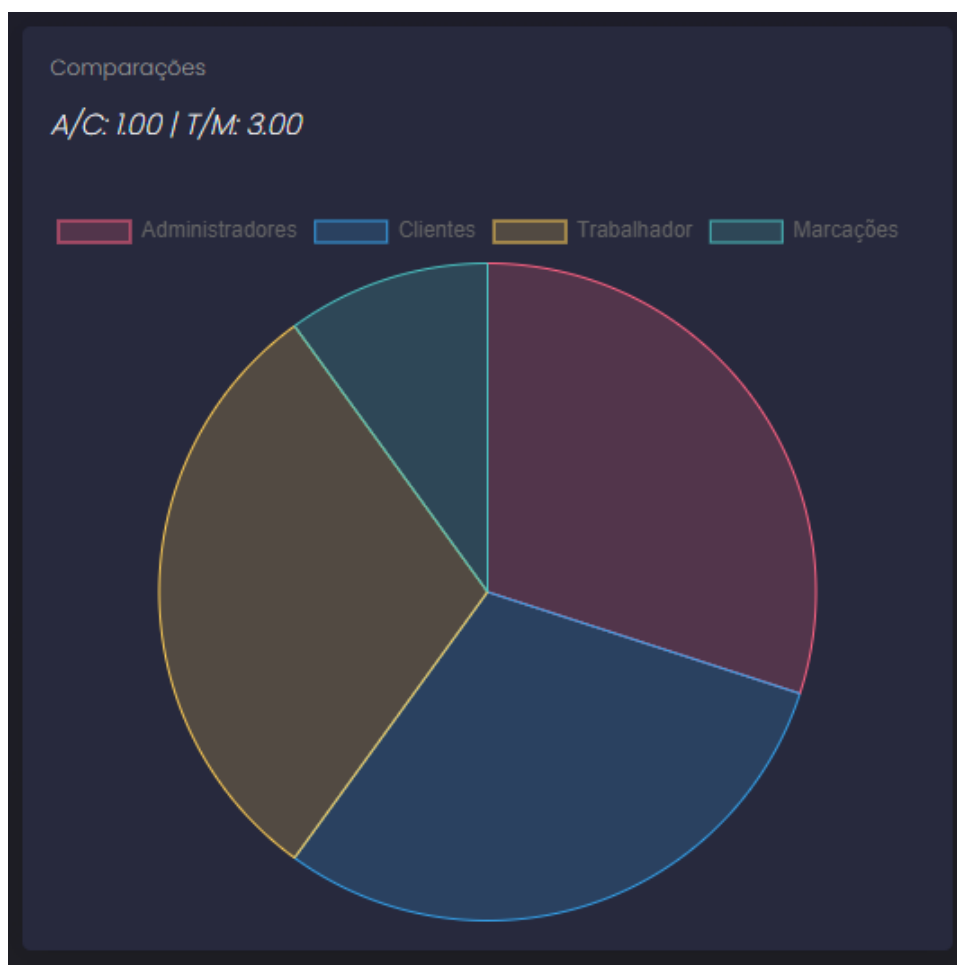


Figura 37- Gráfico de comparações

Fonte: Printscreen tirado do dashboard da página administrador do nosso site

A figura 37 demonstra-nos um gráfico no qual podemos observar várias comparações, como quantos Administradores existem por cliente e também quantos trabalhadores existem por marcação que por sua vez permite apercebermo-nos de uma forma mais fácil se devemos empregar mais ou se devemos despedir alguns trabalhadores como também alguns dos administradores. Ao carregar em uma ou mais labels fará com que as fatias do gráfico correspondentes a essas labels sejam ocultas.

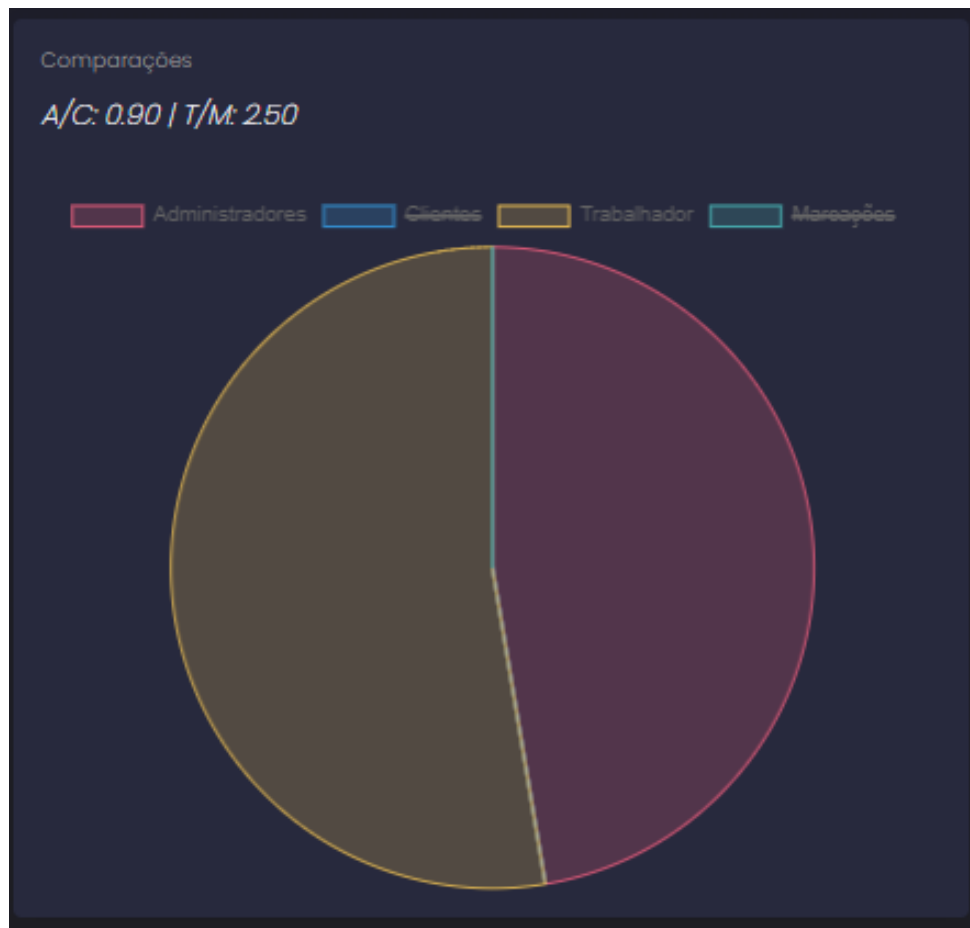


Figura 38- Gráfico de comparações sobre administradores e quantidade de trabalhadores

Fonte: Printscreen tirado do dashboard da página administrador do nosso site

Permitindo fazer comparações visuais como, por exemplo, a figura 38 demonstra a quantidade de administradores comparada a quantidade de trabalhadores.

Para criar o gráfico reutilizamos as funções utilizadas para criar os gráficos anteriores adicionando uma nova função assíncrona para obter o número dos administradores, na função principal, que desenha o gráfico, fizemos chamadas as funções anteriormente referidas com um await para esperar o retorno de cada uma delas antes de prosseguir com a criação do gráfico.

4.1.7. Clientes

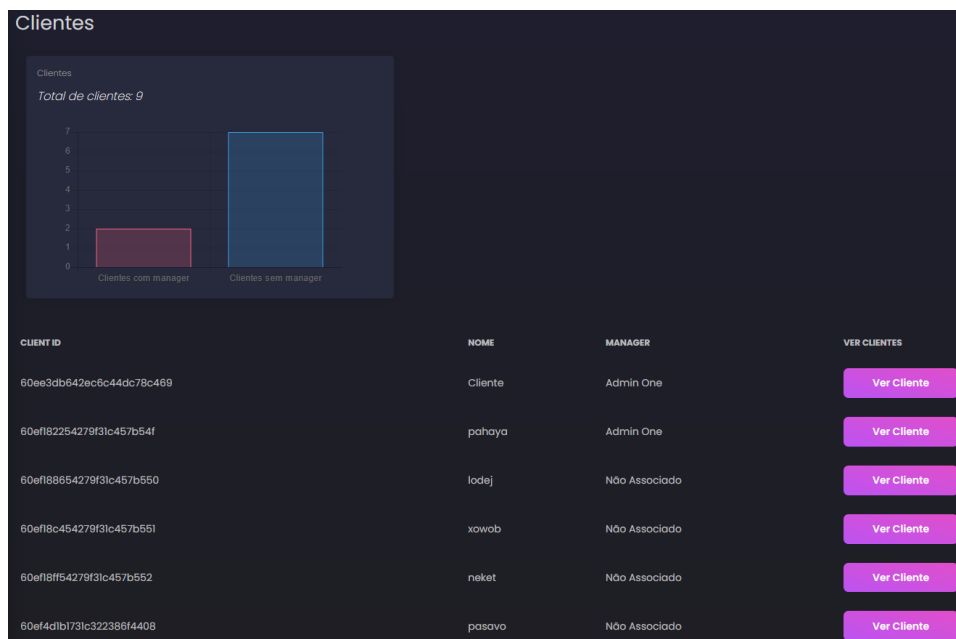


Figura 39- Informações sobre os clientes

Fonte: Printscreen tirado da página clientes só acessada pelos administradores do nosso site.

A página clientes mostrada na Figura 39 tem como finalidade apresentar todos os clientes da empresa e alguns gráficos ilustrativos com informações pertinentes.

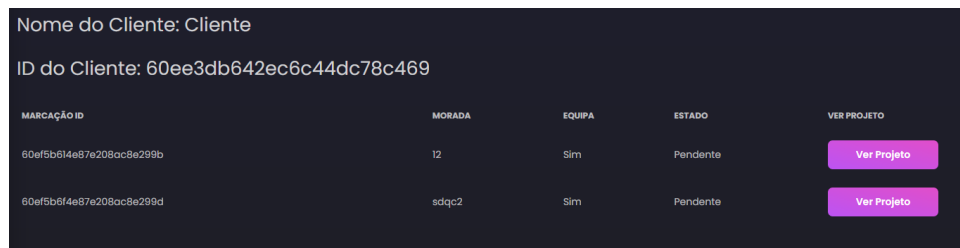
A apresentação dos clientes é feita através da construção dinâmica de uma tabela, que é preenchida através de uma função de back-end, que por sua vez, executa uma query à base de dados. Através de uma função Ajax, é feito um pedido 'GET' à função de back-end. No caso de o pedido ser bem-sucedido, é devolvido, em formato JSON, todos os objetos clientes. A função assíncrona buildTable encarrega-se de preencher cada linha da tabela com um cliente. São adicionadas informações como o clientID, o nome do cliente e o respetivo manager do cliente.

A função buildTable, para além de preencher a tabela com as informações acima referidas, encarrega-se também de adicionar um botão 'Ver Cliente'. Este botão, quando premido, serve para que o administrador possa obter mais informações acerca de um cliente em específico. Utilizando jQuery, foi necessário adicionar ao botão um event handler. Este event handler aguarda que o botão seja premido para poder guardar em sessão, através da função sessionStorage.setItem, o id do cliente específico que o administrador queira ver. De seguida o event handler redireciona o administrador para a página ver_cliente.

4.1.7.1. Ver Cliente

A página `ver_cliente` tem como objetivo obter um conjunto de informações acerca de um cliente em específico, após ter sido encontrado na página anterior.

Com base no id do cliente que foi guardado em sessão, é possível retirá-lo e utilizá-lo para executar uma query à base de dados. A função de back-end encarregou-se de encontrar um objeto cliente com base no id, e a sua resposta é um documento em formato JSON, com todas as informações relativas ao cliente.



Nome do Cliente: Cliente
ID do Cliente: 60ee3db642ec6c44dc78c469

MARCAÇÃO ID	MORADA	EQUIPA	ESTADO	VER PROJETO
60ef5b6f4e87e208ac8e299b	12	Sim	Pendente	<button>Ver Projeto</button>
60ef5b6f4e87e208ac8e299d	sdqc2	Sim	Pendente	<button>Ver Projeto</button>

Figura 40- Informações do cliente

Fonte: Printscreen tirado da página `ver_cliente` só acedida pelos administradores do nosso site.

As informações obtidas vão permitir apresentar informações como o nome do cliente, o id do cliente e as marcações do cliente tal como estão demonstradas na Figura 40. A tabela é semelhante às demais demonstradas e funciona de forma idêntica. A diferença prende-se na coluna equipa, que verifica se uma equipa está atribuída ou não a uma marcação. Caso não esteja, o administrador que se associou a este cliente, poderá adicionar a equipa.

A função `buildTable` adiciona o botão ‘Ver Projeto’ a cada marcação, no qual foi adicionado um event handler. O event handler aguarda, e quando o botão tiver sido premido, é guardado em sessão o id da marcação do cliente para que mais tarde possa ser utilizado. De seguida o event handler redireciona o administrador para a página `ver_projeto`.

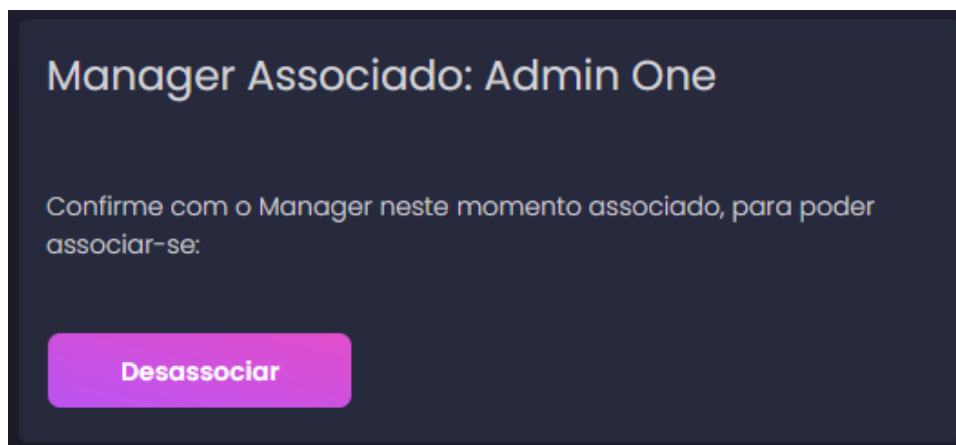


Figura 41- Manager associado ou não

Fonte: Printscreen tirado da página `ver_cliente` depois de ter clicado em associar só acedida pelos administradores do nosso site.

O cartão da Figura 41 tem como objetivo mostrar qual é o administrador que, neste momento está associado ao cliente, ou seja, qual é o manager que faz a gestão deste cliente e das suas limpezas. Foi utilizado jQuery para criar assincronamente o cartão.

Quando um manager está associado, este pode desassociar-se. O botão tem um event handler que aguarda, e quando este for premido é executado um pedido ‘PUT’ para a função de back-end `disAssAdmin()`. Esta função tem dois passos:

- Encontrar, com base no id do objeto cliente que referencia o objeto administrador, e retirar o cliente do array `clients` do administrador;
- Encontrar o cliente com base no seu id e alterar os campos `admin` para null e `hasAdmin` para false;

Quando um manager não está associado, este pode associar-se ao cliente. O botão tem um event handler que aguarda, e quando este for premido é executado um pedido ‘PUT’ para a função de back-end `hasAdmin()`. Esta função segue os mesmos passos que a anterior só que ao contrário.

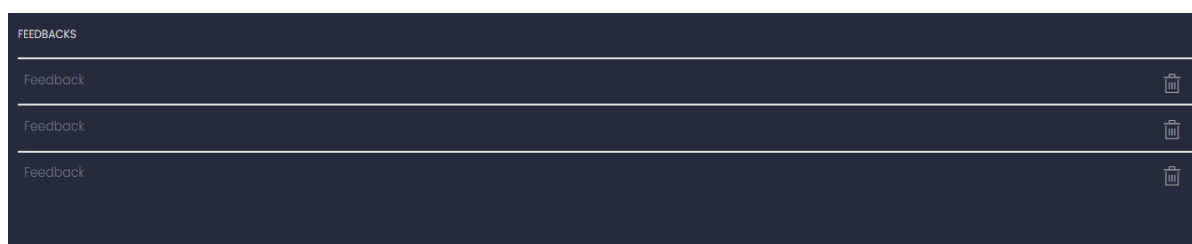


Figura 42- Feedbacks dos clientes

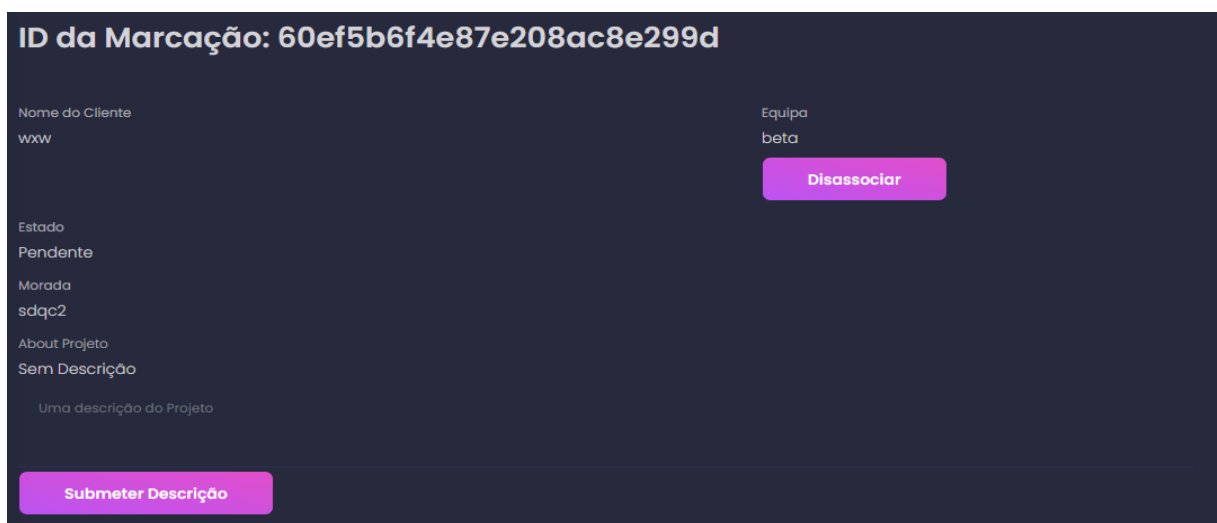
Fonte: Printscreen tirado da página ver_cliente só acedida pelos administradores do nosso site.

Nesta página é possível visualizar uma tabela que contém todos os feedbacks dados por um cliente em específico mostrada na Figura 42. É utilizado a função populate() para devolver um array de feedbacks do cliente. Estes por sua vez são mapeados dinamicamente e assincronamente para uma tabela através de uma função ajax que envia um pedido ‘GET’ juntamente com o id do cliente.

4.1.7.2. Ver Projeto

A página ver_projeto surge quando um administrador prime o botão ‘Ver Projeto’. O id da marcação é retirado da sessão, e é utilizado para fazer queries à base de dados.

Com base neste id é possível obter a marcação de uma limpeza em específico, que está atribuída a um cliente.



The screenshot displays a web form titled 'ID da Marcação: 60ef5b6f4e87e208ac8e299d'. The form is organized into two columns. The left column contains labels and values for 'Nome do Cliente' (wxw), 'Estado' (Pendente), 'Morada' (sdqc2), and 'About Projeto' (Sem Descrição). The right column shows 'Equipa' (beta) and a 'Disassociar' button. At the bottom, there is a text input field labeled 'Uma descrição do Projeto' and a 'Submeter Descrição' button.

Figura 43- Informações da marcação

Fonte: Printscreen tirado da página ver_projeto só acedida pelos administradores do nosso site.

Este cartão de Bootstrap mostrado na Figura 43 é dinamicamente criado de acordo com as informações presentes no objeto marcação. Algumas das informações vão estar presentes durante o decorrer da limpeza enquanto outras não. As informações que vão estar presentes são:

o id da marcação, o nome do cliente, a equipa se tiver atribuída ou não, a morada e uma breve descrição que o utilizador pode acrescentar acerca da marcação.

Porém, os conteúdos do cartão vão variar de acordo com algumas condições:

- No caso desta marcação não tiver sido atribuída a uma equipa, é permitido ao utilizador fazer essa atribuição de acordo com as equipas existentes. A função de atribuição de uma equipa a uma marcação no back-end recebe dois id's e é composta por três passos:
 - Associação através do relacionamento de um para muitos, entre uma marcação e uma equipa. Uma equipa pode ter várias marcações;
 - A alteração do campo equipa para 'Sim' no objeto marcação, significando que a marcação passa a ter uma equipa associada;
 - E por fim, adicionar aos arrays de marcações dos três trabalhadores da equipa, a marcação em questão.
- No caso desta marcação já tiver sido atribuída a uma equipa e que o seu estado ainda esteja em 'Pendente', será possível desassociar a equipa por qualquer razão. A função de back-end que implementa esta funcionalidade segue os mesmos passos, mas de forma inversa, da função de atribuição.
- Ao terminar a limpeza, os campos de aval_client e aval_admin, a avaliação do cliente e a avaliação dos administradores respetivamente, são apresentados ao utilizador. O campo aval_client representa a avaliação que o cliente fez do trabalho realizado e a aval_admin é um campo onde o administrador pode dar uma nota final à marcação para que os trabalhadores da equipa possam estar a par.

4.1.7.3. Trabalhadores

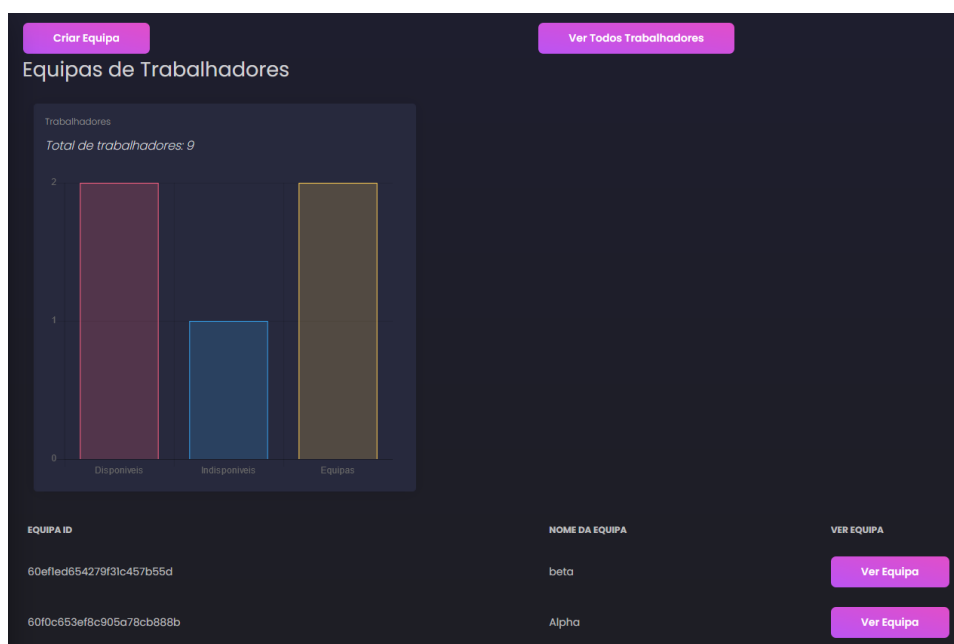


Figura 44- Página dos trabalhadores do back office

Fonte: Printscreen tirado da página de trabalhadores só acessada pelos administradores do nosso site.

A página dos trabalhadores mostrada na figura 44 tem como finalidade fornecer ao administrador da empresa, um ambiente para a análise, manipulação e gestão do recurso humano trabalhador.

O utilizador tem acesso à informação relativa às equipas que estão disponíveis para serem atribuídas a uma marcação; pode criar uma equipa com trabalhadores que estejam disponíveis para trabalhar e pode verificar uma lista com todos os trabalhadores que a empresa emprega.

A tabela da página é construída de forma semelhante às demais demonstradas, sendo ela construída assincronamente através de um pedido 'GET' pela função Ajax. A função de back-end responde ao pedido realizando uma query simples find(), de todos os objetos equipa da coleção Equipas. A resposta é materializada em formato JSON e a função buildTable trata de preencher a tabela com o id de cada equipa e o respetivo nome.

Para além disso, a função buildTable adiciona a cada linha da tabela, um botão 'Ver Equipa', ao qual é adicionado uma função onclick denominada store(). A função store recebe como parâmetro o index i do array de equipas que foi enviado pelo servidor. Este index é utilizado para guardar o id da equipa na sessão do passport.js para mais tarde ser utilizado.

4.1.7.3.1. Ver Equipa

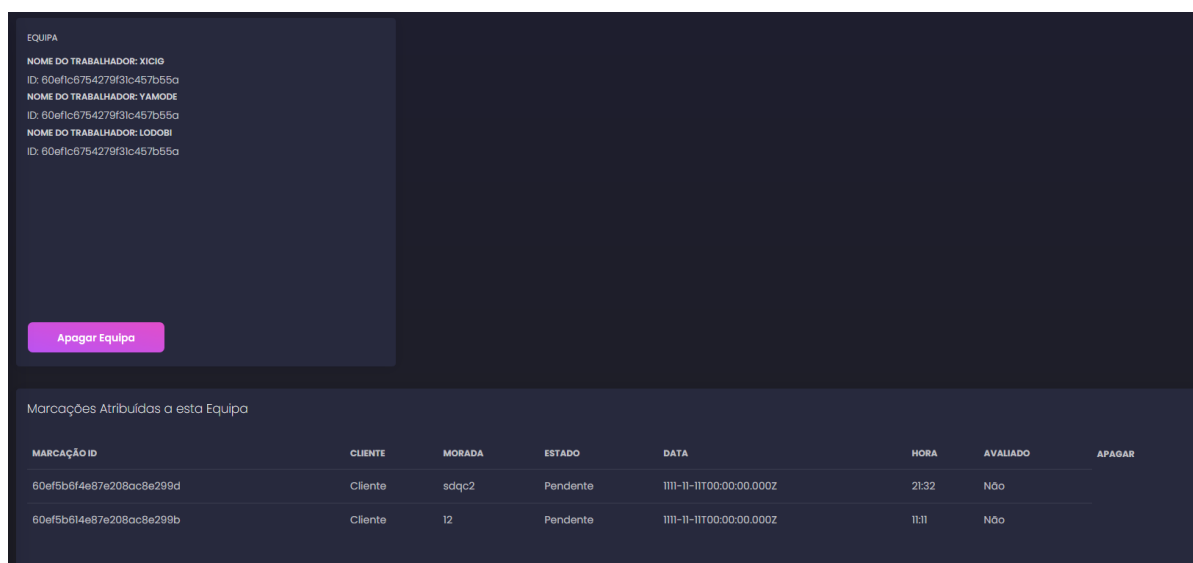


Figura 45- Página de ver equipa

Fonte: Printscreen tirado da página de ver equipa só acedida pelos administradores do nosso site

Esta página representada na Figura 45 caracteriza-se por ter dois cartões Bootstrap: um cartão que apresenta todas as marcações atribuídas a esta equipa, e outro que apresenta os membros (os trabalhadores) que compõe a equipa.

A tabela é construída utilizando uma função ajax que envia um pedido ‘GET’ ao servidor. A função de back-end recebe o id da equipa que foi retirada da sessão do utilizador e constrói queries à base de dados. É utilizada a função populate() para enviar os conteúdos do array das marcações da equipa em formato JSON. Por fim a tabela é preenchida com as seguintes informações: o id da marcação, o nome do cliente, a morada da limpeza, o seu estado, a sua data, a sua hora e um campo que indica se a marcação foi avaliada por um administrador. A última coluna respeita um conjunto de condições para cada marcação:

- No caso de uma marcação estiver apenas terminada e não avaliada, a função buildTable adiciona um botão ‘Avaliar’ que redireciona o utilizador à página ver_projeto para que o administrador possa avaliar o trabalho realizado.
- No caso de uma marcação estiver terminada e avaliada, então a função buildTable adiciona um botão para poder apagá-la.

No cartão que apresenta os trabalhadores que compõem a equipa, é possível apagá-la. Está associado ao botão ‘Apagar Equipa’ um event handler que envia através de uma função

Ajax um pedido de 'DELETE' da equipa. A função de apagar uma equipa toma os seguintes passos:

- Recebe quatro ids nomeadamente o id da equipa, do primeiro trabalhador, do segundo trabalhador e do terceiro trabalhador;
- Encontra uma marcação com base no id da equipa, e altera os campos equipa para “Não”, significando que esta marcação não terá uma equipa associada. Atribui null ao campo team, cujo faz a referência a um objeto equipa. Visto que tal não existe então este campo tem de ser null;
- Encontra todos os trabalhadores por id e altera o campo equipa, deixando a referência de um objeto equipa ser preenchida com null e altera o campo pequipa significando que um trabalhador já não pertence a uma equipa e por isso está disponível para ser escolhido para uma nova equipa ou, que por outras razões, tenha que se retirar da disponibilidade para trabalhar. O array marcTrab é limpo de todas as marcações,
- E por fim é apagado a equipa com base num id.

4.1.7.3.2. Criar Equipa

WORKER ID	NOME
60ef53bb5710e020f4d7865e	dehela
60ef954dbec8791d98e3195a	Diogo
60f0d30cef8c905a78cb888c	Kanguro

Nome da Equipa

Nome:

Selecionar Trabalhadores

Primeiro Trabalhador Segundo Trabalhador Terceiro Trabalhador

Figura 46- Página de criar equipa

Fonte: Printscreen tirado da página de criar equipa só acedida pelos administradores do nosso site

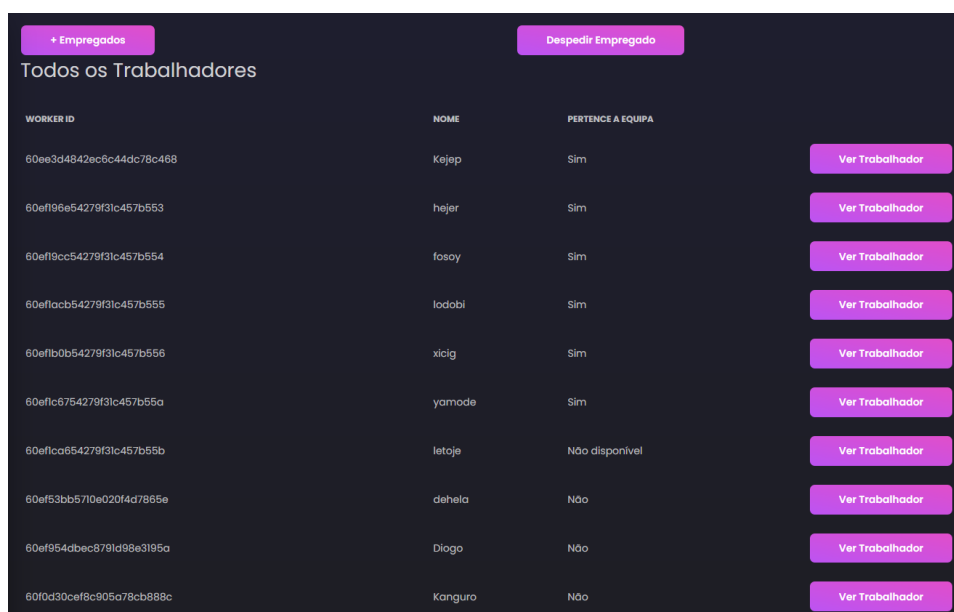
Nesta página mostrada na Figura 46 estão disponibilizados dois setores principais: os trabalhadores disponíveis e a criação de uma equipa.

Nos trabalhadores disponíveis foi criada uma tabela que retorna o *Id* e o nome de todos os trabalhadores que não pertençam a uma equipa. Esta tabela tem como objetivo facilitar a procura de um trabalhador desejado para perceber se esse trabalhador se encontra disponível para se juntar a uma equipa ou se já pertence a uma.

No segundo sector, isto é, o sector da criação de uma equipa, temos presente quatro entradas de dados, em que a primeira é um `<input>` para o nome da equipa, e as três restantes são menus “DropDown” criados através do `<select>`. Estes três menus têm associadas três funções:

- Uma função denominada como *sliders* que utiliza o mesmo *ajax* da criação da tabela, utilizando os mesmos valores, isto é, o *Id* e o nome do trabalhador. Contudo, na tabela os valores são retornados como linha através da *tag* `<tr>` e nos menus “DropDown” são retornados como `<option>` em que o nome é o que está visível na opção, e o *Id* é o “*value*” para que quando os três trabalhadores sejam submetidos numa equipa, estejam disponíveis os *Id*’s de cada um.
- Uma segunda função que bloqueia um trabalhador selecionado nos restantes menus para impedir que uma equipa tenha duas ou três vezes o mesmo funcionário.
- E por fim, a terceira função que através do `<form>` recolhe o nome da equipa e os três trabalhadores selecionados. Com o botão “Criar equipa” tal como indica, faz o *submit* dos quatro valores e gera uma nova equipa na base de dados.

4.1.7.3.3. Ver todos os Trabalhadores



WORKER ID	NOME	PERTENCE A EQUIPA	
60ee3d4842ec5c44dc78c468	Kejep	Sim	Ver Trabalhador
60ef96e54279f31c457b553	hejer	Sim	Ver Trabalhador
60ef9cc54279f31c457b554	fosoy	Sim	Ver Trabalhador
60ef1acb54279f31c457b555	lodobl	Sim	Ver Trabalhador
60ef1b0b54279f31c457b556	xicig	Sim	Ver Trabalhador
60ef1c6754279f31c457b55a	yamode	Sim	Ver Trabalhador
60ef1ca654279f31c457b55b	leteje	Não disponível	Ver Trabalhador
60ef53bb5710e20f4d7865e	dehela	Não	Ver Trabalhador
60ef954dbec8791d98e3195a	Diogo	Não	Ver Trabalhador
60f0d30cef8c905a78cb888c	Kanguro	Não	Ver Trabalhador

Figura 47- Página de ver todos os trabalhadores

Fonte: Printscreen tirado da página de ver todos os trabalhadores só acedida pelos administradores do nosso site

A página `ver_todos_trabalhadores` mostrada na Figura 47 tem como objetivo apresentar uma lista de todos os trabalhadores da empresa. A tabela que mostra os trabalhadores, inclui informações como o id do trabalhador, o seu nome, e um campo que indica se o trabalhador pertence a uma equipa ou não, ou ainda se está disponível ou não. É possível conhecer mais informações sobre um trabalhador em específico ao premir o botão ‘Ver trabalhador’. Esta funcionalidade age nos mesmos moldes que foram descritos para as páginas ‘ver_equipa’, ‘ver_cliente’ e ‘ver_projeto’ nos pontos 4.2.5.3.1, 4.2.5.2.2, 4.2.5.2.1, respetivamente.

Para além das funcionalidades já descritas, a página permite ao utilizador adicionar mais empregados – fazendo um redirect para a página de ‘Registo de Staff’ - e permite despedir (remover) um trabalhador com base no seu id.

4.1.7.4. Projetos / Marcações

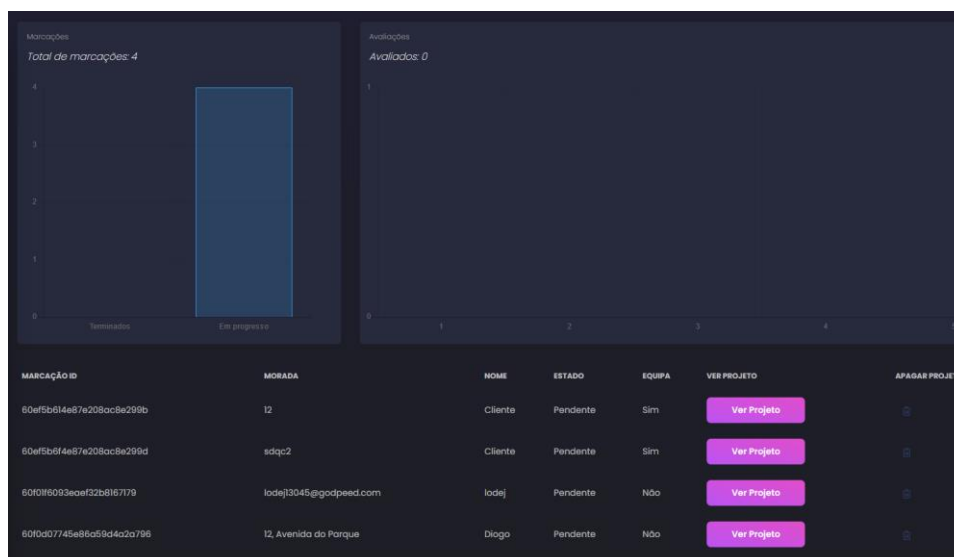


Figura 48- Página projetos/marcações

Fonte: Printscreen tirado da página de projetos/marcações só acedida pelos administradores do nosso site

A página Projetos/Marcações mostrada na Figura 48 tem como objetivo apresentar uma lista de todas as marcações de limpezas submetidas por todos os clientes, bem como alguns gráficos informativos.

Tal como as restantes páginas, esta inclui uma tabela construída dinamicamente com as seguintes informações: os ids das marcações, as suas moradas, os clientes que as submeteram, o seu estado e um campo que verifica se já foi atribuída uma equipa à marcação.

É adicionado dois botões a cada linha da tabela nomeadamente um botão ‘Ver Projeto’, que segue a mesma lógica de implementação das páginas ver_cliente, ver_todos_trabalhadores, ver_trabalhador e ver_equipa acima descritas, e um botão para que o administrador possa apagar a marcação, caso esta tenha terminado ou por qualquer outra razão.

Como nas restantes páginas, foi adicionado uma função onclick que, ao premir o botão ‘Ver Projeto’ de uma certa marcação, guarda o id da marcação para mais tarde ser retirada da sessão e utilizada para procurar na base de dados a marcação específica que o administrador queira ver.

4.1.7.4.1. Ver Projeto

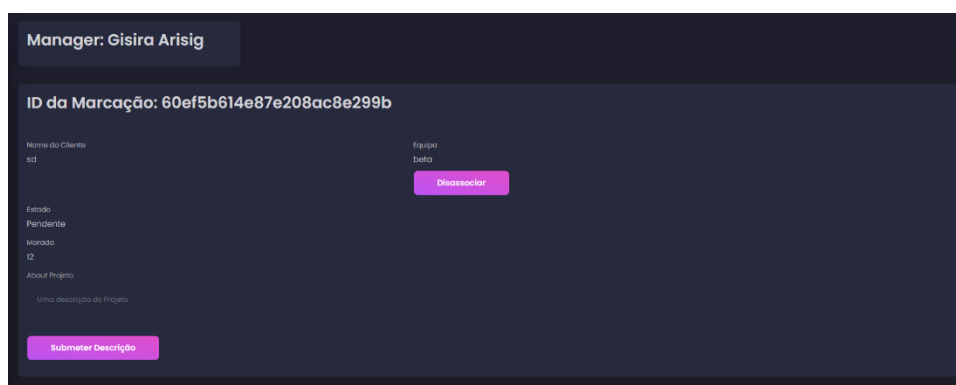


Figura 49- Página ver projeto

Fonte: Printscreen tirado da página de ver projeto só acedida pelos administradores do nosso site

Após premir o botão da página anterior designado por ‘Ver Projeto’, o administrador é redirecionado para a página `ver_projeto_marcacoes` mostrada na Figura 49, diferente da `ver_projeto`, onde poderá conhecer mais detalhes sobre a marcação.

A página é construída nos mesmos moldes que a página `ver_projeto`, ou seja, existe um cartão Bootstrap que recebe informação a tempo real dependendo de algumas condições. O cartão apresenta as mesmas funcionalidades que a página `ver_projeto` ou seja, o administrador do cliente poderá dar uma descrição ao trabalho, associar / desassociar uma equipa e avaliar caso esta tenha a limpeza terminada.

A única diferença entre as duas páginas é que esta não permite ao administrador desassociar-se do cliente que submeteu a marcação. Só é permitido ao administrador do cliente desassociar-se na área do cliente. Esta página tem como única finalidade conhecer mais informações sobre os projetos.

4.1.7.5. Feedbacks

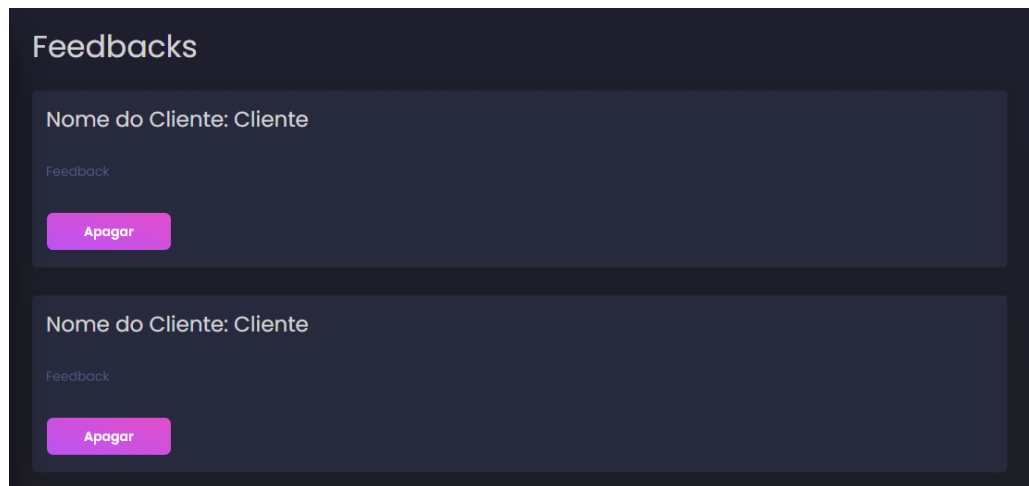


Figura 50- Página feedbacks só acedida pelos administradores

Fonte: Printscreen tirado da página feedback só acedida pelos administradores do nosso site

A página *feedbacks* mostrada na *Figura 50* é utilizada para os administradores poderem ver todas as opiniões referentes aos serviços prestados pelos trabalhadores. Como se pode ver na imagem, cada comentário está dentro de uma “card” e associado a ele está o nome de quem o escreveu e um botão de apagar caso o *feedback* seja inapropriado ou inadequado para as observações da empresa.

Para criar uma “card” é utilizada a função *containers*. Esta função faz o levantamento de todos os *feedbacks* armazenados na base de dados e com o auxílio da função “*for*” percorre cada um e em cada “card” atribui os dados já acima referidos

Através do “*form*” com o *Id* associado a esse *feedback* e o botão “Apagar” é possível fazer o *delete* desse *feedback* na base de dados.

4.1.7.6. Perfil

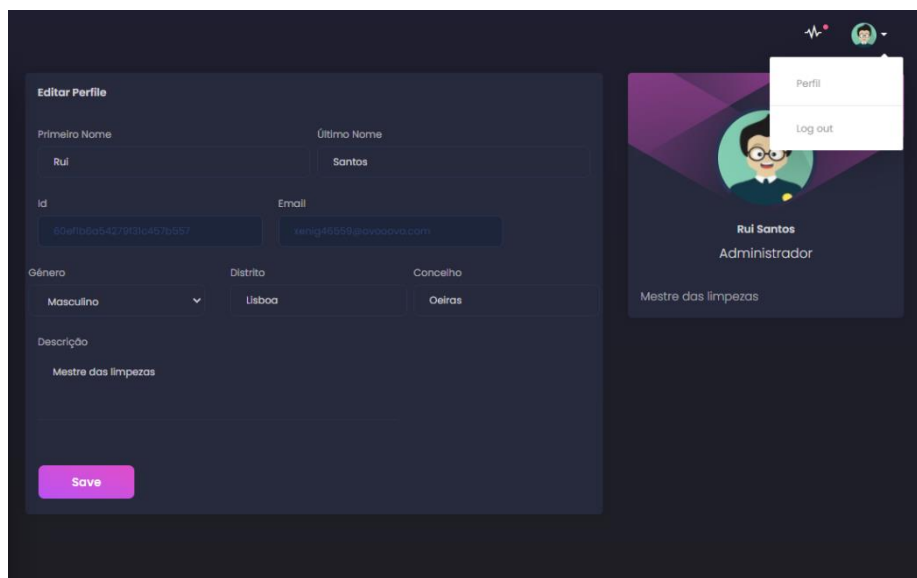


Figura 51- Página do user do administrador

Fonte: Printscreen tirado da página user só acedida pelos administradores do nosso site

Para aceder à página “user” mostrada na Figura 51 o administrador tem que ir ao canto superior direito e seleccionar “Perfil”.

Quando seleccionado, vai aparecer um pequeno cartão com o nome e a descrição do utilizador e um formulário com os dados obrigatórios já preenchidos e novos campos, como o género, distrito, concelho e descrição, por preencher, caso seja essa a vontade do administrador visto que estes campos não são obrigatórios. Aí, também tem disponível o *Id* e o email associados a ele, no entanto, esses valores não podem ser alterados, são só para consulta, se necessário.

Para disponibilizar os dados atuais do administrador, são utilizadas duas funções, a *buildForm* e a *card* para o formulário e para a *card*, respetivamente. A função *card* retorna apenas o nome e a descrição, contudo a função *buildForm* constrói todo o formulário presente na página.

Nesta construção foi incluído o *form* para poder fazer o update dos dados do administrador, através do método “PUT”, caso o utilizador assim o deseje. Assim que o administrador fizer o save das suas alterações, os dados são automaticamente atualizados e passam a estar presentes no formulário.

Conclusões

Neste relatório descrevemos todo o processo da criação da nossa aplicação.

Numa primeira fase apresentamos todas as ferramentas utilizadas neste projeto salientando o uso de Bootstrap HTML e jQuery para a criação do front-end; express.js e node.js para construção do back-end.

Numa segunda fase fizemos uma análise da estrutura da base de dados MongoDB, explicando os relacionamentos entre as várias estruturas de dados. Para além disso analisámos cada página da nossa aplicação web, apontando todas as funcionalidades das mesmas.

Infelizmente, alguns dos objetivos traçados não foram alcançados. Teriam sido momentos de aprendizagem que permitiriam um alargamento dos conhecimentos adquiridos ao longo da licenciatura e na construção deste projeto. Os objetivos que não foram alcançados são: um menu de notificações que alertariam o administrador num eventual acontecimento, a criação do form ‘Contact Us’ que permitiria o cliente enviar um email de esclarecimento para o email da empresa e a construção de uma aplicação de chat *built-in*. Não querendo desperdiçar esta oportunidade de aprendizagem, continuaremos a trabalhar a fim de concluir.

Retirámos que, a combinação das ferramentas utilizadas neste projeto proporcionou-nos um excelente ambiente de trabalho para criarmos a aplicação web. A escolha da base de dados foi fundamental para conseguirmos concluir o projeto dentro das datas estabelecidas.

Dado a natureza da linguagem NoSQL e a dimensão do projeto, o tempo despendido na construção das coleções de dados foi significativamente menor do que se fosse escolhido uma base de dados SQL tradicional. O MongoDB proporcionou um ambiente amigável a várias alterações na lógica de negócio que facilitou em muito o nosso trabalho.

Uma outra conclusão retirada foi a de que a nossa aplicação necessitava de ferramentas como o jQuery e o Ajax para prover um ambiente de execução assíncrono e dinâmico, fundamental à sua consecução. Com algum engenho e arte conseguimos combinar todas as estas ferramentas.

Foi para nós um desafio elaborar um sistema de registo/login seguro e confiável aos seus utilizadores. Após investigação, encontramos uma solução que permitiu separar a lógica dos três tipos diferentes de utilizadores, salvaguardando a sua privacidade.

Concluímos com a certeza que o trabalho efetuado foi ao encontro dos propósitos enunciados.

Bibliografia

- [1] “Universidade Autónoma de Lisboa,” Wikipédia, 6 abril 2019. [Online]. Available: https://pt.wikipedia.org/wiki/Universidade_Aut%C3%B3noma_de_Lisboa. [Accessed: 14-Jul-2021].
- [2] “JavaScript,” Wikipedia, 11-May-2021. [Online]. Available: <https://pt.wikipedia.org/wiki/JavaScript>. [Accessed: 14-Jul-2021].
- [3] C. Noletto, “Javascript: o que é, aplicação e como aprender a linguagem JS,” Blog da Trybe, 06-Jul-2021. [Online]. Available: <https://blog.betrybe.com/javascript/>. [Accessed: 14-Jul-2021].
- [4] M. D. N. contributors, “HTML: HyperText Markup Language,” MDN. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML>. [Accessed: 16-Jul-2021]
- [5] Y. Pacievitch, “HTML - Informática,” InfoEscola. [Online]. Available: <https://www.infoescola.com/informatica/html/>. [Accessed: 14-Jul-2021].
- [6] Y. Pacievitch, “Cascading Style Sheets (CSS) - Informática,” InfoEscola. [Online]. Available: <https://www.infoescola.com/informatica/cascading-style-sheets-css/>. [Accessed: 16-Jul-2021].
- [7] M. D. N. contributors, “HTML: HyperText Markup Language,” MDN. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTML>. [Accessed: 16-Jul-2021]
- [8] “Bootstrap (front-end framework),” Wikipedia, 08-Jul-2021. [Online]. Available: [https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)). [Accessed: 16-Jul-2021].

- [9] S. Norton, “Bootstrap 5: What's New About It and Release Date,” Designmodo, 14-Jun-2021. [Online]. Available: <https://designmodo.com/bootstrap-5/>. [Accessed: 16-Jul-2021].
- [10] “Node.js,” Wikipedia, 24-May-2021. [Online]. Available: <https://pt.wikipedia.org/wiki/Node.js>. [Accessed: 16-Jul-2021].
- [11] “Express.js,” Wikipedia, 12-Jun-2021. [Online]. Available: <https://en.wikipedia.org/wiki/Express.js>. [Accessed: 16-Jul-2021].
- [12] M. D. N. contributors, “Express/Node introduction - Learn web development: MDN,” Learn web development | MDN. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/Introduction. [Accessed: 16-Jul-2021].
- [13] D. Wilson, “express-session,” npm. [Online]. Available: <https://www.npmjs.com/package/express-session>. [Accessed: 16-Jul-2021].
- [14] “MongoDB,” Wikipedia, 07-Feb-2020. [Online]. Available: <https://pt.wikipedia.org/wiki/MongoDB>. [Accessed: 16-Jul-2021].
- [15] R. Dakar, “Romulo Dakar,” Desenvolvedor Ninja, 06-Dec-2018. [Online]. Available: <http://desenvolvedor.ninja/mongodb-o-que-e-e-para-que-serve/>. [Accessed: 16-Jul-2021].
- [16] N. Karnik, “Introduction to Mongoose for MongoDB,” freeCodeCamp.org, 07-Jun-2019. [Online]. Available: <https://www.freecodecamp.org/news/introduction-to-mongoose-for-mongodb-d2a7aa593c57/>. [Accessed: 16-Jul-2021].
- [17] J. Hanson, “passport,” npm. [Online]. Available: <https://www.npmjs.com/package/passport>. [Accessed: 16-Jul-2021].
- [18] “Documentation,” Passport.js. [Online]. Available: <http://www.passportjs.org/docs/>. [Accessed: 16-Jul-2021].

- [19] J. Hanson, “passport-local,” Passport.js, 01-Jan-1967. [Online]. Available: <http://www.passportjs.org/packages/passport-local/>. [Accessed: 16-Jul-2021].
- [20] “Bcrypt,” Wikipedia, 06-Jul-2021. [Online]. Available: <https://en.wikipedia.org/wiki/Bcrypt>. [Accessed: 16-Jul-2021].
- [21] jQ. contributors, jQuery. [Online]. Available: <https://jquery.com/>. [Accessed: 16-Jul-2021].
- [22] “jQuery,” Wikipedia, 17-Jun-2021. [Online]. Available: <https://en.wikipedia.org/wiki/JQuery>. [Accessed: 16-Jul-2021].
- [23] “Ajax (programming),” Wikipedia, 22-May-2021. [Online]. Available: [https://en.wikipedia.org/wiki/Ajax_\(programming\)#Technologies](https://en.wikipedia.org/wiki/Ajax_(programming)#Technologies). [Accessed: 16-Jul-2021].
- [24] “Introducing JSON,” JSON. [Online]. Available: <https://www.json.org/json-en.html>. [Accessed: 16-Jul-2021].
- [25] “JSON,” Wikipedia, 16-Jul-2021. [Online]. Available: <https://en.wikipedia.org/wiki/JSON>. [Accessed: 16-Jul-2021].
- [26] jwt auth0.com, “JSON Web Tokens Introduction,” JSON Web Token Introduction. [Online]. Available: <https://jwt.io/introduction>. [Accessed: 16-Jul-2021].
- [27] A. Reinman, Nodemailer. [Online]. Available: <https://nodemailer.com/about/>. [Accessed: 16-Jul-2021].
- [28] J. Hanson and T. J. Holowaychuk, “connect-flash,” npm. [Online]. Available: <https://www.npmjs.com/package/connect-flash>. [Accessed: 16-Jul-2021].
- [29] “Chart.js,” Wikipedia, 11-Mar-2021. [Online]. Available: <https://en.wikipedia.org/wiki/Chart.js>. [Accessed: 16-Jul-2021].

- [30] C. contributors, “Chart.js,” Chart.js | Open source HTML5 Charts for your website. [Online]. Available: <https://www.chartjs.org/>. [Accessed: 16-Jul-2021].
- [31] “Canvas (HTML5),” Wikipedia, 16-Apr-2020. [Online]. Available: [https://pt.wikipedia.org/wiki/Canvas_\(HTML5\)](https://pt.wikipedia.org/wiki/Canvas_(HTML5)). [Accessed: 16-Jul-2021].
- [32] D. Pavlutin, “How to Use Fetch with async/await,” *Dmitri Pavlutin Blog*, 18-Feb-2021. [Online]. Available: <https://dmitripavlutin.com/javascript-fetch-async-await/>. [Accessed: 16-Jul-2021].
- [33] MDN contributors, “Response.json() - Web APIs: MDN,” *Web APIs / MDN*, 04-Jul-2021. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/API/Response/json>. [Accessed: 16-Jul-2021].