



Faculdade de Ciências Exatas e Engenharia



NUDGEWEB

Curso: Engenharia Informática

Disciplina: Estágio/Projeto

Coordenadora: Ana Karina Caldeira Caraban

Filipe Magno de Gouveia Quintal

Alunos: José Alejandro Ferreira Gouveia nº 2028616

Funchal, 23 de julho de 2021

Índice

1. Introdução	3
1.1. Problema.....	3
1.2. Solução.....	3
1.3. Organização do documento	3
2. Tecnologia escolhida	4
2.1. Vantagens	4
2.2. Desvantagens.....	4
2.3. Tipos de notificações	4
2.4. Exemplos de notificações	4
3. Requisitos	6
3.1. Requisitos funcionais.....	6
4. Ferramentas utilizadas	7
4.1. Firebase Cloud Firestore.....	7
4.2. Firebase Authentication	7
4.3. Firebase Cloud Storage.....	7
4.4. Extensões Google Chrome.....	7
4.5. Visual Studio Code.....	7
4.6. Linguagem HTML	7
4.7. Linguagem JavaScript	8
4.8. Vue.js.....	8
5. Diagramas.....	9
5.1. Diagramas Swimlane para login na extensão e na dashboard	9
5.2. Diagrama Swimlane para inicialização dos temporizadores.....	10
5.4. Diagrama Swimlane para editar uma notificação	11
5.5. Diagrama Swimlane para eliminar uma notificação	12
5.6. Diagrama Swimlane para aceder os dados do utilizador	13
6. Desenvolvimento	15
6.1. Extensão Google Chrome	15
6.1.1. Login com ligação a base de dados	15
6.1.2. Registo de websites visitados	18
6.1.3. Registo de interatividade e Top 3 mais interagidos	20
6.1.4. Registo das notificações	22
6.2. Dashboard do administrador	25

6.2.1.	Login	25
6.2.2.	Utilizadores	26
6.2.3.	Notificações	29
6.2.3.1.	Editar notificação	30
6.2.3.2.	Eliminar notificação	32
7.	Conclusão	34
8.	Referências	35

1. Introdução

Para este projeto foi proposto o desenvolvimento de um software com o objetivo de notificar os seus utilizadores durante a sua navegação no browser.

A vantagem da utilização deste software por parte do utilizador prende-se com o facto de que o mesmo consegue ter um maior controlo e perceção do tempo despendido em cada website. Ao obter este controlo poderá obter inúmeros benefícios sejam eles: aumento da produtividade; maior foco na tarefa em detrimento de potenciais distrações; redução da fadiga provocada pelo prolongamento da exposição à “luz azul” do ecrã; maior atenção ao risco de plágio aquando da realização de pesquisas, projetos, dissertações, teses, entre outros, devido ao facto de se poder saber quanto tempo já se está num determinado website.

1.1. Problema

Foi realizado um estudo pelo *Department of Human Factors Engineering*, na Coreia, acerca dos efeitos da curvatura da tela e da duração da tarefa no desempenho da revisão de projetos; desconforto visual; fadiga visual; carga de trabalho mental e satisfação do utilizador e tal estudo chegou à conclusão de que intervalos curtos podem ser eficazes no aumento do desempenho de determinada tarefa e previnem a degradação da saúde ocular.

1.2. Solução

Sendo assim foi desenvolvida uma extensão Google Chrome em que o seu propósito é notificar o utilizador quando está a utilizar um browser durante demasiado tempo para que possa fazer uma pausa de forma que quando voltar da pausa possa aumentar o desempenho nas suas tarefas.[1]

1.3. Organização do documento

No próximo capítulo apresenta-se as “Notificações” - que será a ferramenta principal deste trabalho e que funcionará como solução para o problema apresentado no parágrafo anterior. O referido capítulo apresentará ainda dois tipos de notificações bem como alguns exemplos de notificações. No terceiro capítulo explora-se o tema dos “Requisitos”, dando principal destaque aos requisitos funcionais que representam essencialmente o que o software faz em termos de tarefas e serviços. O quarto tema está relacionado com as “Ferramentas Utilizadas” onde se refere todas as linguagens de programação, aplicações, serviços e base de dados utilizadas no decorrer deste projeto, tais como: Firebase Cloud Firestore, Firebase Authentication, Firebase Cloud Storage, Extensões Google Chrome, Visual Studio Code, Linguagem HTML, Linguagem JavaScript e Vue.js. O quinto capítulo do projeto relaciona-se com o “Desenvolvimento” e refere os passos dados no decorrer do mesmo. No primeiro passo foi importante distinguir o que é uma extensão Google Chrome, como contruir a mesma e as respetivas funcionalidades e no segundo passo procurou-se contruir a dashboard do administrador - que representa a interface gráfica para que o administrador possa aceder e editar os dados na base de dados. Por último apresentar-se-á a “Conclusão” do trabalho onde discutir-se-á todos os pontos desenvolvidos de uma forma mais geral e se a solução escolhida foi ideal para a resolução do problema.

2. Tecnologia escolhida

A tecnologia escolhida para este projeto foram as notificações. As notificações funcionam como alertas visuais e/ou auditivos emitidos por aplicações ou serviços que têm como objetivo avisar o utilizador de algo.

Atualmente, as notificações são um recurso importante nos smartphones, mas que também podem ser implementadas noutras tecnologias, como por exemplo, computadores, tablets e até mesmo relógios (Smartwatches).

Num estudo com 15 participantes, Pielot et al. Investigou-se como é que os utilizadores interagem com as notificações. Ao longo de uma semana, os participantes receberam uma média de 63.5 notificações por dia, sendo grande parte de aplicações de mensagens e emails.[2]

2.1. Vantagens

A vantagem das notificações é que as aplicações e serviços utilizam-na para informar os utilizadores proactivamente sobre vários eventos, tais como, novas mensagens, emails, atualizações e eventos relacionados com as aplicações em questão.[3]

2.2. Desvantagens

O uso intensivo de notificações também pode produzir efeitos negativos, tais como, distrações, interrupções e aumento de stress. Uma equipa de investigadores investigou várias formas para reduzir estes efeitos negativos, como, adiar notificações para pontos de interrupção/intervalos, usar dados de contexto para avaliar a interruptibilidade do utilizador ou filtrar as notificações indesejadas.[3]

2.3. Tipos de notificações

Existem dois tipos de notificações: notificação Push e notificação Pull, a primeira também conhecida como server push notification, é a entrega de informação de um aplicativo para um dispositivo sem que tenha sido feito um pedido específico do utilizador. Este tipo de notificação é utilizado nos smartphones atuais, tendo sido introduzida pela Apple em 2009 com o nome Apple Push Notification service (APNs) com o objetivo dos desenvolvedores de aplicativos transmitirem informação para os dispositivos iOS, watchOS, tvOS e macOS. Em 2010, a Google lançou a sua própria versão do serviço para os dispositivos Android com o nome Google Cloud to Device Messaging (C2DM). Normalmente o utilizador decide se pretende ativar os alertas durante o processo de instalação da aplicação, caso o utilizador opte por ativar, será possível gerir o funcionamento dos alertas.[4]

O segundo tipo de notificação – notificação Pull – corresponde às atualizações entregues a um computador ou smartphone em resposta ao utilizador ou sondagem iniciada por um software de um servidor remoto. Este tipo de notificações entregam as atualizações não-críticas e as comunicações. Estas geralmente dependem do software existente instalado no computador ou smartphone.[5]

2.4. Exemplos de notificações

Atualmente, existem diversas aplicações que utilizam as notificações push, tais como:

- **Facebook** – A app envia notificações de gostos e comentários das publicações do utilizador e em publicações que o próprio seja identificado e também envia um aviso dos aniversários dos seus amigos.
- **Warframe** – Envia notificações das missões disponíveis no jogo.
- **Twitter** – Envia notificações dos novos tweets das contas que o utilizador segue.
- **Twitch** - Envia notificações quando os streamers que o utilizador segue iniciam uma livestream.

3. Requisitos

Nesta secção serão abordados os requisitos necessários de modo a se conseguir alcançar a solução desejada. Tais requisitos foram escolhidos durante uma reunião em conjunto com a coordenadora responsável, onde foram discutidos quais os requisitos mais apropriados para o desenvolvimento deste projeto.

3.1. Requisitos funcionais

Os requisitos funcionais são os requisitos que descrevem o comportamento do sistema em condições específicas. Representam as funcionalidades que devem ser implementadas pelos desenvolvedores, de forma que os utilizadores consigam efetuar as suas tarefas.

Os requisitos funcionais deste sistema são:

RF-01. O sistema deverá permitir um utilizador efetuar login utilizando um sistema de autenticação.

RF-02. O sistema deverá permitir um administrador efetuar login utilizando um sistema de autenticação.

RF-03. O sistema deverá efetuar verificação do tipo de entidade que está a iniciar sessão através da base de dados e efetuar o reencaminhando para o seu devido menu.

RF-04. O sistema deverá criar e registar como novo utilizador na base de dados caso a entidade no qual esteja a iniciar sessão não se encontre presente na base de dados.

RF-05. O sistema deverá registar na base de dados os websites visitados durante uma sessão do browser do utilizador.

RF-06. O sistema deverá registar na base de dados o tempo total em que o website esteve aberto.

RF-07. O sistema deverá registar na base de dados o número de interatividade feita no website pelo utilizador

RF-08. O sistema deverá registar na base de dados os 3 websites com maior interatividade do utilizador.

RF-09. O sistema deverá registar na base de dados as notificações criadas pelo administrador.

RF-10. O sistema deverá permitir ao administrador editar e eliminar as notificações existentes.

RF-11. O sistema deverá permitir ao administrador visualizar os 3 websites com maior interatividade de cada utilizador.

RF-12. O sistema deverá permitir ao administrador alterar o tipo de prioridade do utilizador.

RF-13. O sistema deverá permitir ao administrador visualizar os websites visitados por cada utilizador.

4. Ferramentas utilizadas

A seleção de ferramentas utilizadas foi escolhida devido à sua compatibilidade com o projeto a ser desenvolvido.

4.1. Firebase Cloud Firestore

A Cloud Firestore é uma base de dados de documentos NoSQL que permite armazenamento, sincronização e consulta de dados, mantém os seus dados em sincronia em aplicações cliente através de listeners em tempo real e oferece suporte off-line para dispositivos móveis e Web para que seja possível criar aplicações responsivas que funcionem independentemente da latência da rede ou da conectividade com a internet.[6], [7]

4.2. Firebase Authentication

O Firebase Authentication é uma ferramenta que fornece facilidade e segurança no desenvolvimento de um sistema de autenticação seguro com compatibilidade com diversos Login API's como por exemplo, Google, Facebook, Twitter, entre outros.[8]

4.3. Firebase Cloud Storage

O Cloud Storage é um serviço de armazenamento de objetos avançado. Ao utilizar os SDKs do Firebase é possível usar a segurança do Google para fazer upload e download de ficheiros, independentemente da qualidade da rede.[9]

4.4. Extensões Google Chrome

As extensões Google Chrome são programas que podem ser instalados no browser Google Chrome de forma a alterar a funcionalidade do browser. Isto inclui adicionar novas funcionalidades ao browser ou modificar o comportamento existente do browser de forma a ficar mais conveniente para o utilizador.[10]

Exemplos de extensões existentes:

- **Adblock** – bloqueia a apresentação de anúncios.
- **Ghostery** – bloqueia cookies e plug-ins indesejáveis.
- **Honey** – encontra automaticamente códigos de cupões de mais de 100 lojas online.

4.5. Visual Studio Code

O Visual Studio Code é um editor de código, que vem com um suporte incorporado para JavaScript, TypeScript e Node.js e ainda tem acesso a uma variedade de extensões para outras linguagens como, C++, C#, Java, Python, Go.[11]

4.6. Linguagem HTML

HTML (HyperText Markup Language) é a linguagem de programação utilizada para a construção de páginas Web sendo assim o construtor de blocos mais básico da Web e define o significado e a estrutura do conteúdo Web. Além do HTML, utilizam-se outras tecnologias no desenvolvimento das páginas Web, geralmente é usado o CSS para descrever a aparência da página e o JavaScript para a funcionalidade.[12]

4.7. Linguagem JavaScript

JavaScript é uma linguagem de programação que permite aos seus utilizadores implementar diversas funcionalidades mais complexas em páginas Web. Permitindo assim a página apresentar mais do que informação estática, como por exemplo, pode mostrar em tempo real conteúdos atualizados, mapas interativos, animações gráficas em 2D/3D, vídeos, etc.[13]

4.8. Vue.js

O Vue.js é uma framework progressiva que permite o desenvolvimento de interfaces de utilizador e aplicações de página única.[14]

5. Diagramas

Para representar visualmente o funcionamento da extensão e da dashboard, foram utilizados os seguintes diagramas.

Os **Diagramas Swimlane para login na extensão e na dashboard** apresentados abaixo representam o processo de efetuar login do utilizador e do administrador na extensão e na dashboard sendo que na dashboard tem umas pequenas diferenças.

5.1. Diagramas Swimlane para login na extensão e na dashboard

Na extensão, o utilizador/administrador efetua o login, o sistema verifica se os dados já existem na base de dados, se não estiverem, é criada uma nova conta como utilizador.

Já na dashboard ao efetuar o login, o sistema verifica se os dados introduzidos correspondem a algum administrador, se não corresponderem, o utilizador é reencaminhado para uma página de aviso.

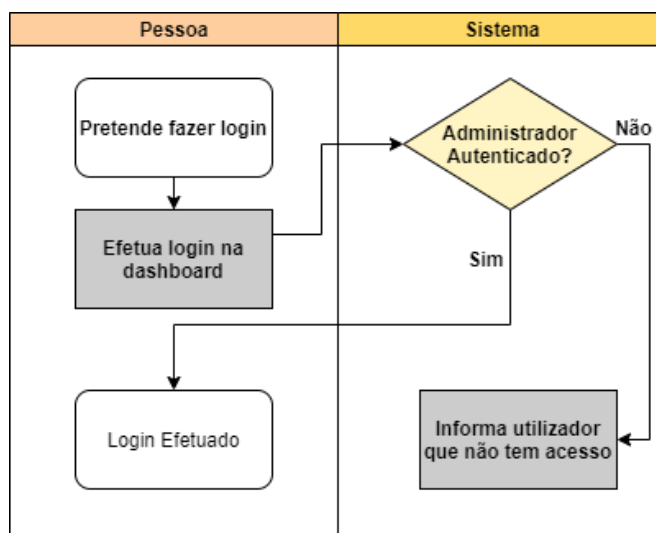


Figura 1 - Diagrama Swimlane para login na dashboard

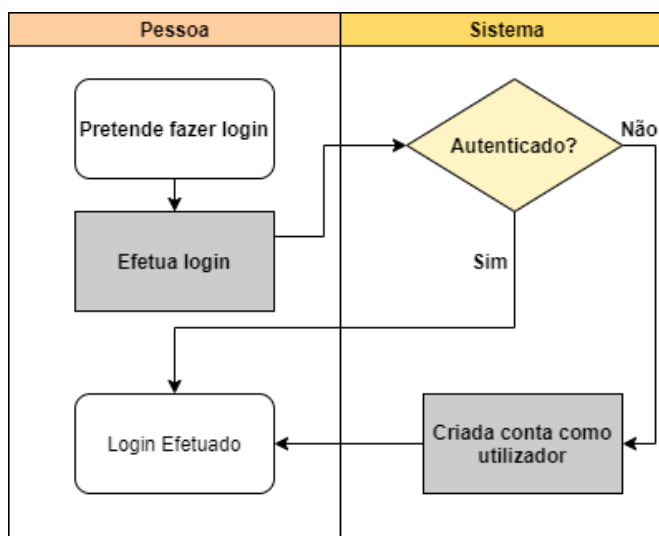


Figura 2 - Diagrama Swimlane para login na extensão

5.2. Diagrama Swimlane para inicialização dos temporizadores

Este diagrama swimlane representa o funcionamento dos temporizadores (Tempo total gasto no browser e tempo total gasto num website específico) e sua interação com a base de dados, que são inicializados após o utilizador efetuar login.

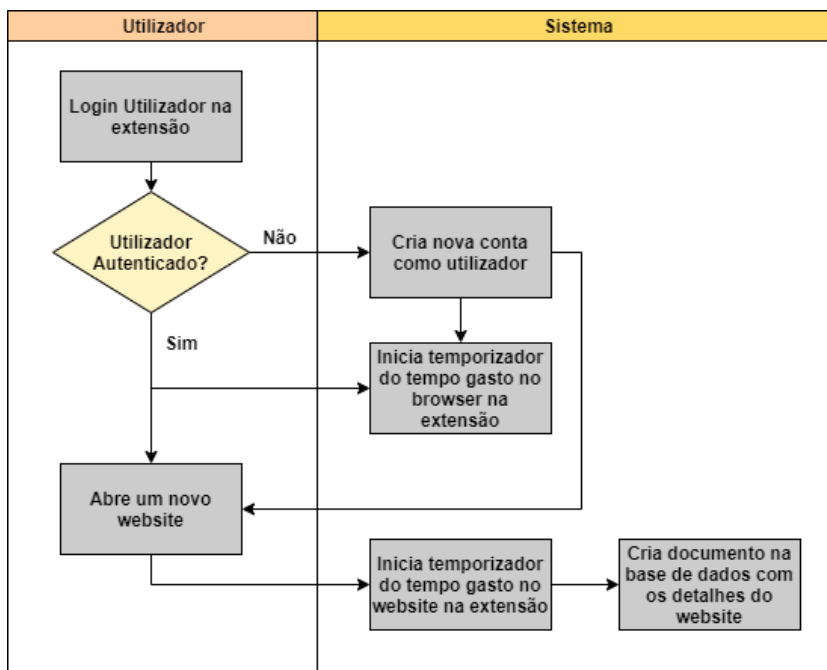


Figura 3 - Diagrama Swimlane para inicializar os temporizadores

5.3. Diagrama Swimlane para criar uma notificação

O seguinte diagrama swimlane demonstra o processo de criação de uma notificação após um administrador efetuar o login na extensão.

Durante o processo são feitas duas validações, a primeira verifica se todos os campos estão preenchidos e a segunda verifica se o ID introduzido já existe na base de dados, caso um dos campos não esteja preenchido e/ou o ID introduzido já exista na base de dados, o sistema informa o administrador, caso esteja tudo preenchido corretamente, a notificação é criada e guardada na base de dados.

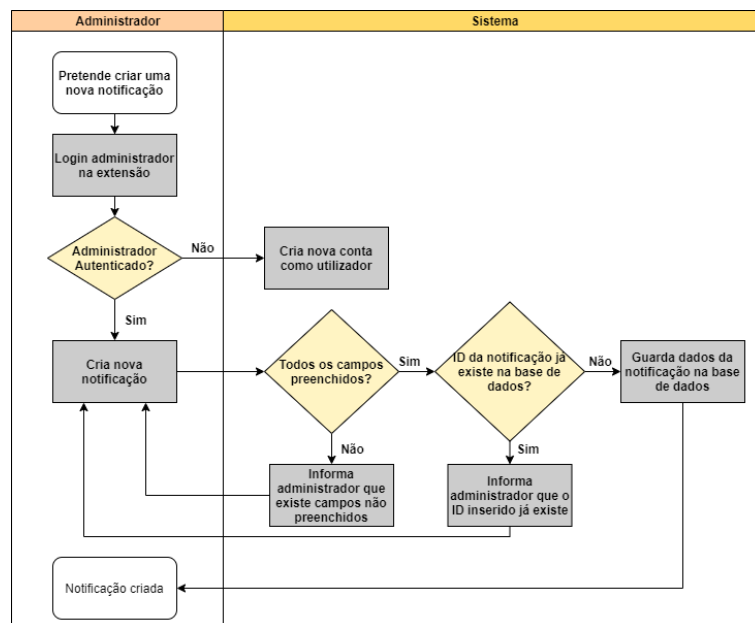


Figura 4 - Diagrama Swimlane para criar uma notificação

5.4. Diagrama Swimlane para editar uma notificação

Neste diagrama swimlane é apresentada, a edição de notificações, uma das funcionalidades da dashboard ao visualizar uma notificação existente.

Para este processo é necessário ter o login efetuado, que como foi apresentando anteriormente apenas pode ser feito pelo administrador, após efetuar o login, é selecionada a notificação que pretende editar.

Ao aceder a notificação é possível editar as informações existentes como por exemplo, o título, a mensagem, os itens (no caso da notificação “List”), o seu ícone e imagem (no caso da notificação (Image)). Ao editar-mos a notificação é necessário ter em atenção em não deixar nenhum campo vazio, caso isso aconteça será enviada um aviso de erro para o administrador, se tudo estiver preenchido de forma correta, a notificação será atualizada.

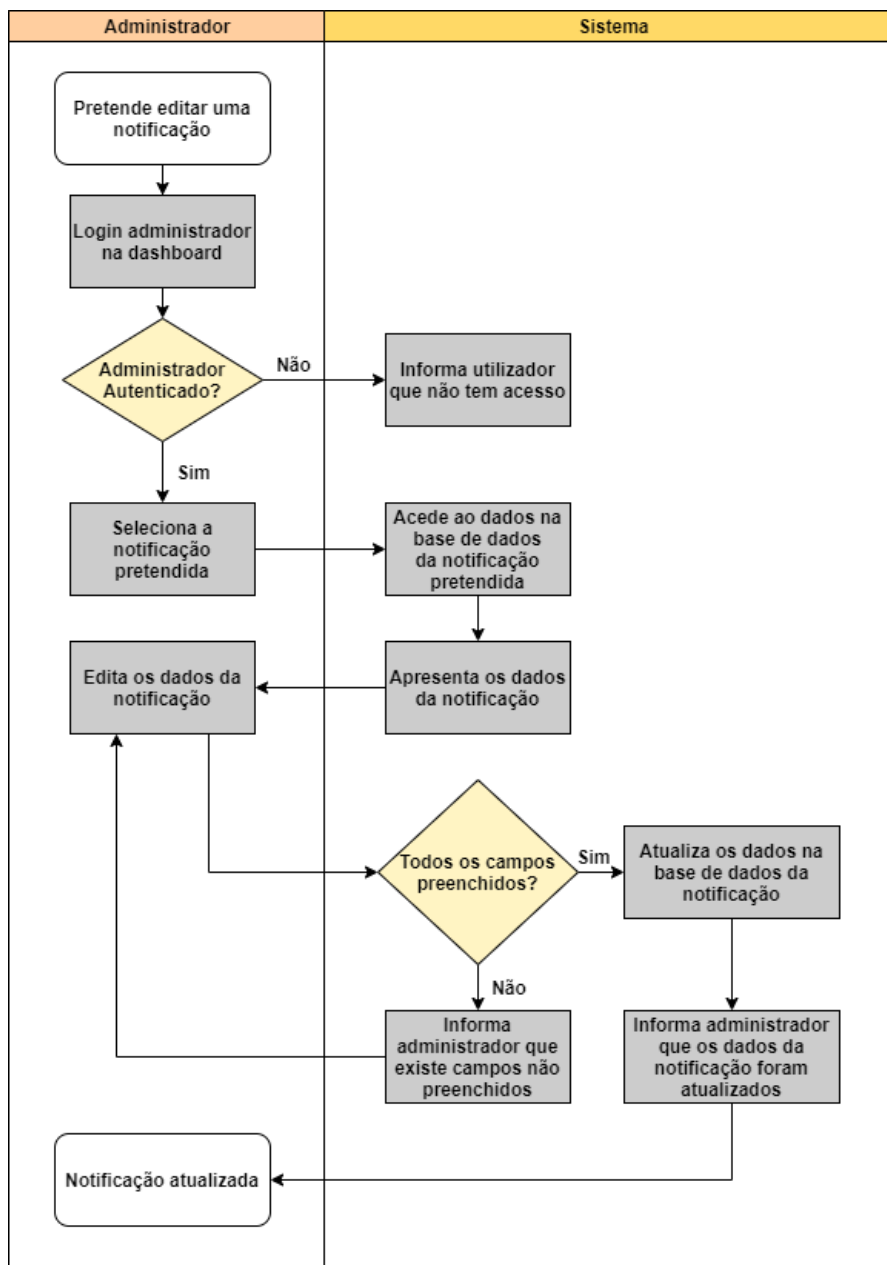


Figura 5 - Diagrama Swimlane para editar uma notificação

5.5. Diagrama Swimlane para eliminar uma notificação

Neste diagrama Swimlane é apresentada, a segunda funcionalidade da dashboard ao visualizar uma notificação existente.

O processo inicial é exatamente o mesmo que o processo para editar uma notificação.

Ao aceder a notificação, caso o administrador sinta que já não existe necessidade para existência daquela notificação, clica no botão “Eliminar”, ao eliminar a notificação, os seus dados são eliminados da base de dados.

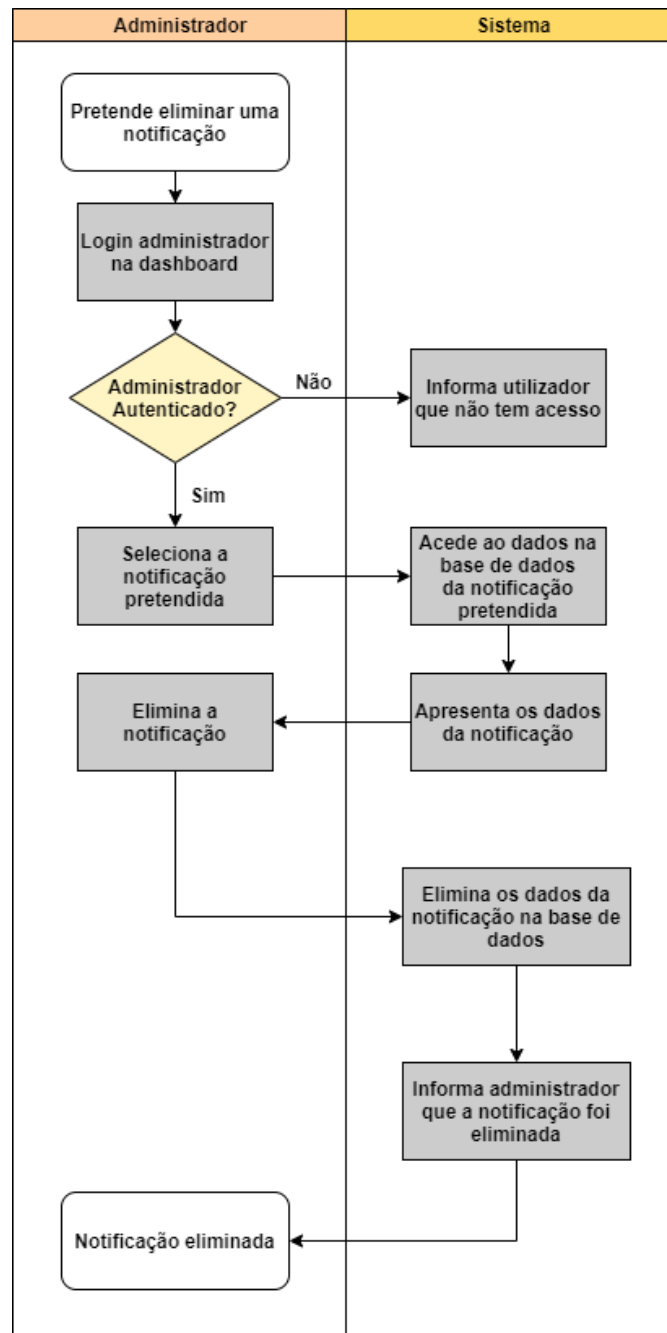


Figura 6 - Diagrama Swimlane para eliminar uma notificação

5.6. Diagrama Swimlane para aceder os dados do utilizador

Neste diagrama Swimlane é representado o processo para aceder os dados do utilizador e para alterar o seu tipo de prioridade.

Ao efetuar login na dashboard será apresentado uma lista dos utilizadores que já utilizaram a extensão, ao seleccionar um utilizador, será apresentado os dados do utilizador, como, o top 3 websites mais interagidos, os detalhes dos websites que foram acedidos e o seu tipo de prioridade, que pode ser alterado.

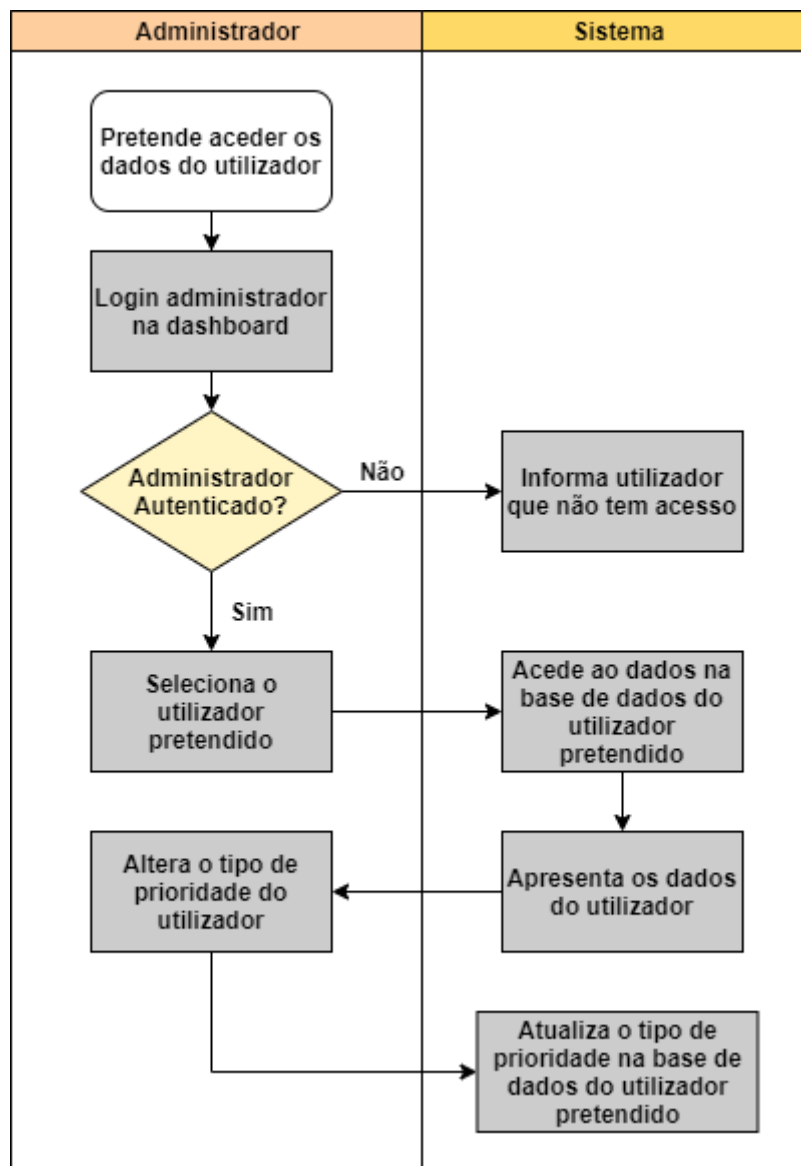


Figura 7 - Diagrama Swimlane para aceder os dados do utilizador via dashboard

6. Desenvolvimento

Neste capítulo serão apresentadas as técnicas utilizadas na construção deste projeto e as respectivas etapas do mesmo. Em primeiro lugar o foco prendeu-se com a extensão Google Chrome e para tal começou-se por estabelecer a ligação do login à base de dados e em seguida foi implementado o registo de diversos dados do utilizador e das notificações na base de dados. Após a conclusão da extensão foi alterado o foco para encontrar uma maneira do administrador ter acesso aos dados de uma forma mais prática e simples e para tal, a solução encontrada foi o desenvolvimento de uma Dashboard.

6.1. Extensão Google Chrome

6.1.1. Login com ligação a base de dados

Uma das funcionalidades mais importantes na extensão foi a criação de um login com ligação à base de dados, pois desta forma é possível registar os utilizadores e diferenciar um utilizador de um administrador durante o seu processo, sendo assim foram utilizados o Firebase Firestore Database e o Firebase Authentication.

Utilizando a documentação referente a esta ferramenta foi implementado o Login da seguinte forma:

```
1 // popup-init.js
2 const firebaseConfig = {
3   apiKey: "AIzaSyDwBgqvF8k158x3ifSh58PKcUOEGGRpPFM",
4   authDomain: "webnot-bc22d.firebaseio.com",
5   databaseURL: "https://webnot-bc22d-default-rtdb.firebaseio.com",
6   projectId: "webnot-bc22d",
7   storageBucket: "webnot-bc22d.appspot.com",
8   messagingSenderId: "348923196506",
9   appId: "1:348923196506:web:661b4625b646f61bd95463",
10  measurementId: "G-6HH867QH88"
11 };
12 // Initialize Firebase
13 firebase.initializeApp(firebaseConfig);
14
15 var db = firebase.firestore();
16
17 function init() {
18   chrome.runtime.sendMessage({ message: 'is_admin_signed_in' },
19     function (response) {
20       if (response.message === 'success' && response.payload) {
21         window.location.replace('../html/admin.html');
22       }
23     });
24   chrome.runtime.sendMessage({ message: 'is_user_signed_in' },
25     function (response) {
26       if (response.message === 'success' && response.payload) {
27         window.location.replace('../html/user.html');
28       }
29     });
30 }
31
32 init();
```

Figura 8 - popup-init.js (Envia mensagem para verificar se já existe alguma sessão iniciada)

```
1 let user_signed_in = false;
2 let admin_signed_in = false;
3
4 chrome.runtime.onMessage.addListener(function (request, sender, sendResponse) {
5   if (request.message === 'is_admin_signed_in') {
6     sendResponse({
7       message: 'success',
8       payload: admin_signed_in
9     });
10  } else if (request.message === 'is_user_signed_in') {
11    sendResponse({
12      message: 'success',
13      payload: user_signed_in
14    });
15  } else if (request.message === 'sign_out_admin') {
16    admin_signed_in = false;
17    sendResponse({ message: 'success' });
18  } else if (request.message === 'sign_out_user') {
19    user_signed_in = false;
20    sendResponse({ message: 'success' });
21  } else if (request.message === 'sign_in_admin') {
22    admin_signed_in = true;
23    sendResponse({ message: 'success' });
24  } else if (request.message === 'sign_in_user') {
25    user_signed_in = true;
26    userLoggedIn();
27    sendResponse({ message: 'success' });
28  }
29
30  return true;
31 });
```

Figura 9 - background.js (Envia mensagem dependendo da mensagem que está a receber)

Ao abrir a extensão, é executado o código **popup-init.js**.

É feita a verificação para ver se já existe um utilizador ou administrador com sessão efetuada, para isso é enviada uma mensagem para o **background.js** que verifica e envia a mensagem *sucess* e se o utilizador ou administrador já têm a sessão efetuada (*true* ou *false*)

Visto que a extensão está a ser aberta pela primeira vez o **background.js** coloca por defeito os `user_signed_in` e `admin_signed_in = false`, ficando assim a aguardar que o utilizador efetue o login.

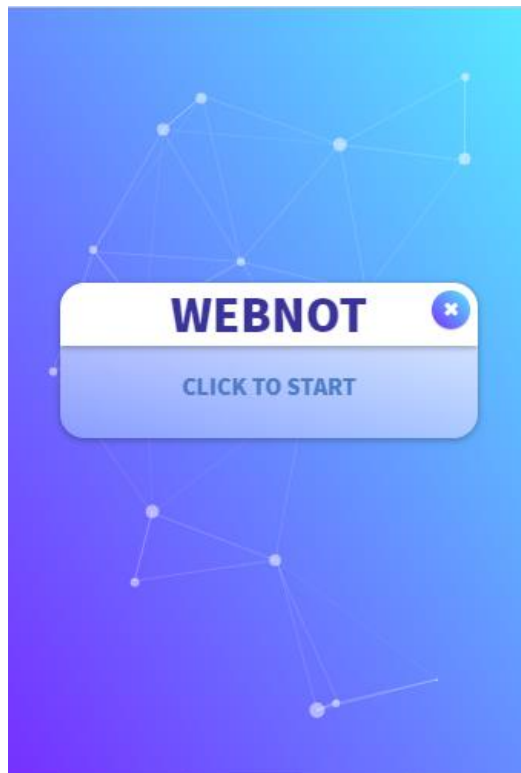


Figura 10 - WebNot Start

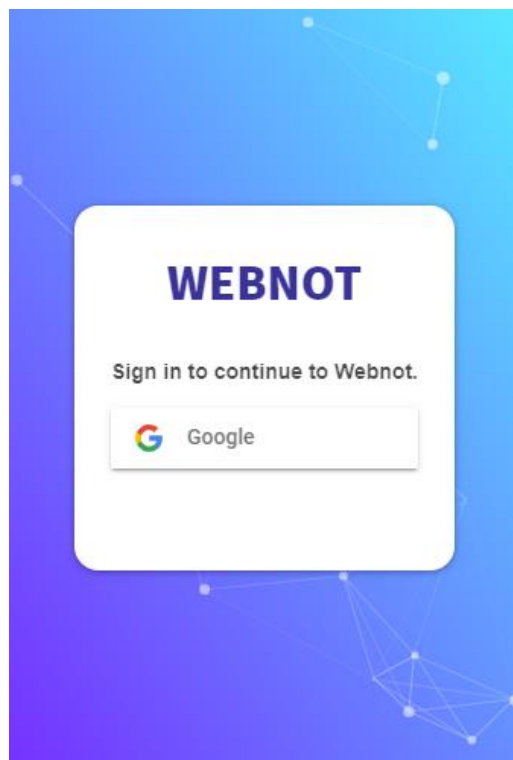


Figura 11 - WebNot Login

```

1 // Init
2 const u
3
4 const u
5   cal
6
7
8
9
10
11
12
13
14   }, sig
15   sig
16
17
18
19
20
21
22
23   },
24 };
25 document
26   ui.
27 })
28
29 function
30   var
31   if
32   use
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51








```

```

53     }).catch((error) => {
54     });
55 });
56 }
57 }
58
59 <function verificaUtilizador(profile) {
60     var docRef = db.collection("user").doc(profile.email);
61     docRef.get().then((doc) => {
62         if (doc.exists) {
63             chrome.storage.local.clear(function() {});
64             window.location.replace('../html/user.html');
65         } else {
66             db.collection("user").doc(profile.email).set({
67                 tipo: "1"
68             })
69             .then(function() {
70                 console.log("Documento escrito com sucesso!");
71                 window.location.replace('../html/user.html');
72             })
73             .catch(function(error) {
74                 console.error("Erro ao escrever o documento: ", error);
75             });
76             db.collection("user").doc(profile.email).collection("dados_websites").doc("top").
77             set({
78                 top1: 0,
79                 top2: 0,
80                 top3: 0,
81                 url_top1: "",
82                 url_top2: "",
83                 url_top3: ""
84             })
85             .then(function() {
86                 chrome.storage.local.clear(function() {});
87                 window.location.replace('../html/user.html');
88             })
89             .catch(function(error) {
90                 console.error("Erro ao escrever o documento: ", error);
91             });
92         }
93     }).catch((error) => {
94     });
95 }

```

Figura 12 - popup-script.js (Autenticador de login, verificaAdmin())

 > admin > josefgouveia9@...		
 webnot-bc22d	 admin  	 josefgouveia9@gmail.com 
+ Iniciar coleção	+ Adicionar documento	+ Iniciar coleção
admin >	josefgouveia9@gmail.com >	+ Adicionar campo
notificacoes	testetestatestando51@gmail.com	tipo: '0'
notificacoesID		
user		

17

O **popup-script.js** é responsável por fornecer quais as opções de login disponíveis que neste caso é apenas o Login Google representado da linha 14-19 e verificação para diferenciar um administrador de um utilizador.

Na função *verificaAdmin()* é efetuada uma ligação à base de dados para verificar se o documento com os dados do utilizador que pretende efetuar login existe e se consta na coleção “Admin” da base de dados. Se existir é enviada uma mensagem para o **background.js** para indicar que o administrador está a fazer login e que deve alterar o *admin_signed_in = true*, após está alteração a extensão altera a sua página login para página do administrador.

Caso o utilizador não conste na coleção “Admin” da base de dados é executada a função *verificaUtilizador()*, que faz basicamente o mesmo que a função *verificaAdmin()* só que no caso deste utilizador já existir na coleção “User”, é feita a alteração na variável *user_signed_in = true* e após isso é reencaminhado para a página do utilizador.

Caso este utilizador não conste em nenhuma das coleções, o próprio é registado como utilizador definindo o seu tipo de utilizador como 1 e é criada o documento “top” com o seu top 3 de sites mais navegados.

6.1.2. Registo de websites visitados

O objetivo desta funcionalidade é o registo dos websites visitados na base de dados juntamente com a data no qual é acedido e o tempo total com a página aberta, para isso foi necessário implementar as seguintes funções no ficheiro **content.js**.

```
306 //Verifica se a página estava apenas minimizada ou se estava fechada, caso estivesse fechada
307 //é iniciado um novo temporizador dessa página e criado um novo documento na base de dados a
308 //partir da funcao guardaDados()
309 function lastPage(){
310     chrome.storage.sync.get("profile",function(res) {
311         db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.
312             hostname).collection("datas").orderBy("data","desc").limit(1).get().then((querySnapshot)
313             => {
314                 querySnapshot.forEach((doc2) => {
315                     if(doc2.exists){
316                         var dataSegundosBD = doc2.data().data_segundos;
317                         var dataAtual = Math.round(Date.now() / 1000);
318                         var resultado = dataAtual-dataSegundosBD < 5;
319                         if(inativo == false){
320                             if(resultado == true){
321                                 mesmaPagina = true;
322                             }
323                             else{
324                                 timeSpentOnSite = 0;
325                                 localStorage.setItem('timeSpentOnSite',timeSpentOnSite);
326                                 mesmaPagina = false;
327                             }
328                         }
329                     } else{
330                         timeSpentOnSite = 0;
331                         localStorage.setItem('timeSpentOnSite',timeSpentOnSite);
332                         mesmaPagina = false;
333                     }
334                 });
335             });
336     });
}
```

Figura 15 - Função lastPage() de content.js

```

455 //Guarda os dados da página actualmente aberta pelo utilizador
456 function guardaDados() {
457   chrome.storage.sync.get("profile",function(res) {
458     var docRef = db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.
459       hostname);
460     docRef.get().then((doc) => {
461       if (doc.exists) {
462         if (mesmaPagina == false){
463           timeSpentOnSite = 0;
464           localStorage.setItem('timeSpentOnSite',timeSpentOnSite);
465           mesmaPagina = true;
466           db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname).
467             collection("datas").doc(new Date().toISOString().slice(0, 10).replace("T"," ") + " " + new Date().
468               toISOString().slice(11, 19).replace("T"," ")).set({
469               full_website: window.location.href,
470               tempo_total_browser_aberto: localStorage.getItem('timeout'),
471               data_segundos: Math.round(Date.now() / 1000),
472               data: new Date().toISOString().slice(0, 10).replace("T"," ") + " " + new Date().toISOString().slice
473                 (11, 19).replace("T"," ")
474             });
475           .then(function() {
476             .catch(function(error) {
477             });
478           });
479         } else {
480           db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname).
481             collection("datas").orderBy("data","desc").limit(1).get().then((querySnapshot) => {
482               querySnapshot.forEach((doc2) => {
483                 db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname).
484                   collection("datas").doc(doc2.id).set({
485                     full_website: doc2.data().full_website,
486                     tempo_total_browser_aberto: localStorage.getItem('timeout'),
487                     data_segundos: Math.round(Date.now() / 1000),
488                     data: doc2.data().data
489                   });
490                 .then(function() {
491                   .catch(function(error) {
492                   });
493                 });
494               });
495             });
496           });
497         }
498       }
499     });
500   });
501 }

```

Figura 16 - Função *guardaDados()* de *content.js*

A função *lastPage()* é, quando o utilizador fecha o website e volta a abrir o mesmo website é sinalizado ao *guardaDados()* utilizado a variável *mesmaPagina* que deve ser criado um novo documento na base de dados indicando que o utilizador está a fazer uma nova pesquisa no website e quando o utilizador apenas minimiza a página ou muda de separador, é sinalizado ao *guardaDados()* utilizado a variável *inativo* que não é necessário criar um novo documento mas que deve sim continuar a escrever informação no ultimo documento existente.

Já a função *guardaDados()* como o próprio nome indica, é guardar os dados do website (URL completo, tempo total com a página aberta, data e a data em segundos) que está a ser visitado no momento quando certas variáveis o permitem. Ao iniciar uma nova página, a variável *mesmaPagina* é iniciada como *false* para indicar que é uma nova página por isso deve ser escrito um novo documento, após isso a variável é colocado a *true* e o controle da escrita no documento criado fica dependente das ações feitas no *lastPage()*.

Ficando depois assim apresentado na base de dados:

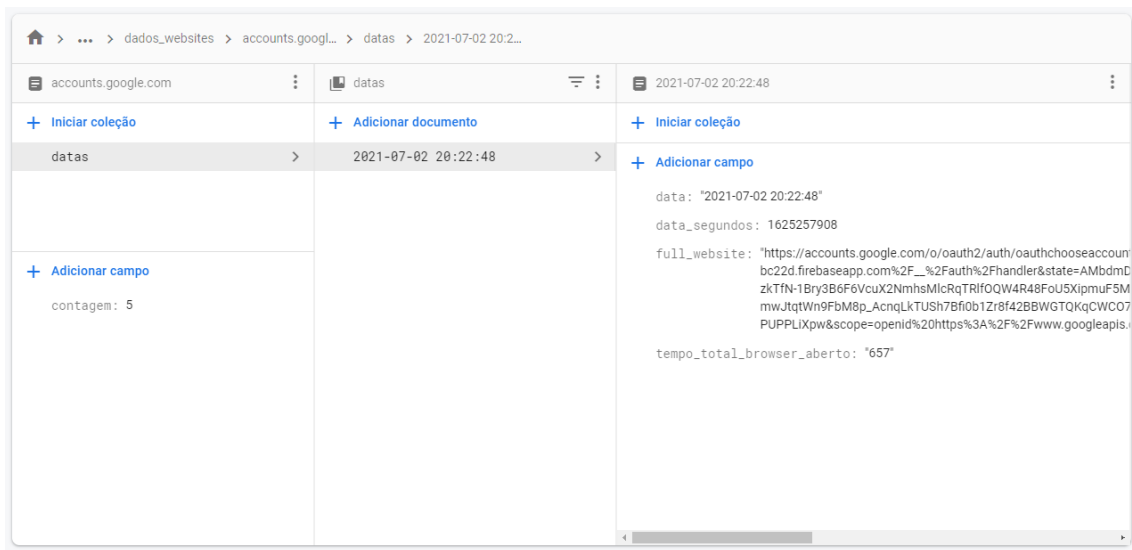


Figura 17 - Firestore (Detalhes do website)

6.1.3. Registo de interatividade e Top 3 mais interagidos

As funções deste sub-tópico, é registar na base de dados de cada website visitado o número de interações feitas, de forma a depois perceber quais os 3 websites mais interagidos pelo utilizador.

```

408 //Conta o numero de interações no website
409 function conta(){
410   chrome.storage.sync.get("profile",function(res) {
411     db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname).get().then((doc) => {
412       if (doc.exists) {
413         var contar = doc.data().contagem;
414         var docRef = db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname);
415         docRef.get().then((doc) => {
416           if (doc.exists) {
417             db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname).set({
418               contagem: contar + 1
419             })
420             .then(function() {
421               })
422             .catch(function(error) {
423               console.error("Erro ao escrever o documento: ", error);
424             });
425           } else {
426             })}.catch((error) => {
427               });
428             });
429           } else {
430             db.collection("user").doc(res.profile).collection("dados_websites").doc(window.location.hostname).set({
431               contagem: 1
432             })
433             .then(function() {
434               })
435             .catch(function(error) {
436               console.error("Erro ao escrever o documento: ", error);
437             });
438           });
439         });
440       }).catch((error) => {
441         });
442       });
443     }

```

Figura 18 - Função conta() de content.js

```

33 //Listener que permite verificar se o hostname da página visitada mudou:
34 chrome.tabs.onUpdated.addListener(function (tabId, changeInfo, tab) {
35   if (changeInfo.status === 'complete') {
36     chrome.tabs.sendMessage(tabId, {type: 'getDoc'}, function (doc) {
37     });
38   }
39 });

```

```

chrome.runtime.onMessage.addListener(
  function(message, sender, sendResponse) {
    switch(message.type) {
      case "getDoc":
        sendResponse(top3());
        sendResponse(conta());
        sendResponse(guardaDados());
        break;

```

Figura 19 - Listener de background.js

Figura 20 - Listener de content.js

```

338 //Regista na base de dados o top 3 páginas mais interagidas
339 function top3(){
340   chrome.storage.sync.get("profile",function(res) {
341     db.collection("user").doc(res.profile).collection("dados_websites").doc("top").get().then((doc) => {
342       var top1;
343       var top2;
344       var top3;
345       var url_top1;
346       var url_top2;
347       var url_top3;
348       top1 = doc.data().top1;
349       top2 = doc.data().top2;
350       top3 = doc.data().top3;
351       url_top1 = doc.data().url_top1;
352       url_top2 = doc.data().url_top2;
353       url_top3 = doc.data().url_top3;
354       if(url_top3 == url_top2){
355         url_top3 = "";
356         top3 = 0;
357       }
358       db.collection("user").doc(res.profile).collection("dados_websites").get().then((querySnapshot) => {
359         querySnapshot.forEach((doc2) => {
360           if(doc2.data().contagem > top1){
361             var docRef = db.collection("user").doc(res.profile).collection("dados_websites").doc("top");
362             docRef.update({
363               top1: doc2.data().contagem,
364               url_top1: doc2.id
365             })
366             .then(function() {
367             })
368             .catch(function(error) {
369               console.error("Erro ao escrever o documento: ", error);
370             });
371             url_top1 = doc2.id;
372             top1 = doc2.data().contagem;
373           }
374           else if((doc2.data().contagem > top2) && (doc2.id != url_top1)){
375             var docRef = db.collection("user").doc(res.profile).collection("dados_websites").doc("top");
376             docRef.update({
377               top2: doc2.data().contagem,
378               url_top2: doc2.id
379             })
380             .then(function() {
381             })
382             .catch(function(error) {
383               console.error("Erro ao escrever o documento: ", error);
384             });
385             url_top2 = doc2.id;
386             top2 = doc2.data().contagem;
387           }
388           else if((doc2.data().contagem > top3) && (doc2.id != url_top1) && (doc2.id != url_top2)){
389             var docRef = db.collection("user").doc(res.profile).collection("dados_websites").doc("top");
390             docRef.update({
391               top3: doc2.data().contagem,
392               url_top3: doc2.id
393             })
394             .then(function() {
395             })
396             .catch(function(error) {
397               console.error("Erro ao escrever o documento: ", error);
398             });
399             url_top3 = doc2.id;
400             top3 = doc2.data().contagem;
401           }
402         });
403       });
404     });
405   });
406 }

```

Figura 21 - Função top3() de content.js

Para isto, é criado um listener no **background.js** que envia uma mensagem para o listener no **content.js** quando é feito uma alteração no hostname do browser indicando assim que foi feita alguma interação sendo assim ela depois registada na base de dados utilizando o *conta()*.

Quanto a função top3(), também é ativada no momento que é feito um update na página sendo feito uma verificação do número de interações de todas as páginas já registadas na base de dados ligada ao utilizador com sessão iniciada e registado/atualizado os 3 websites mais interagidos, como apresentado na imagem abaixo.

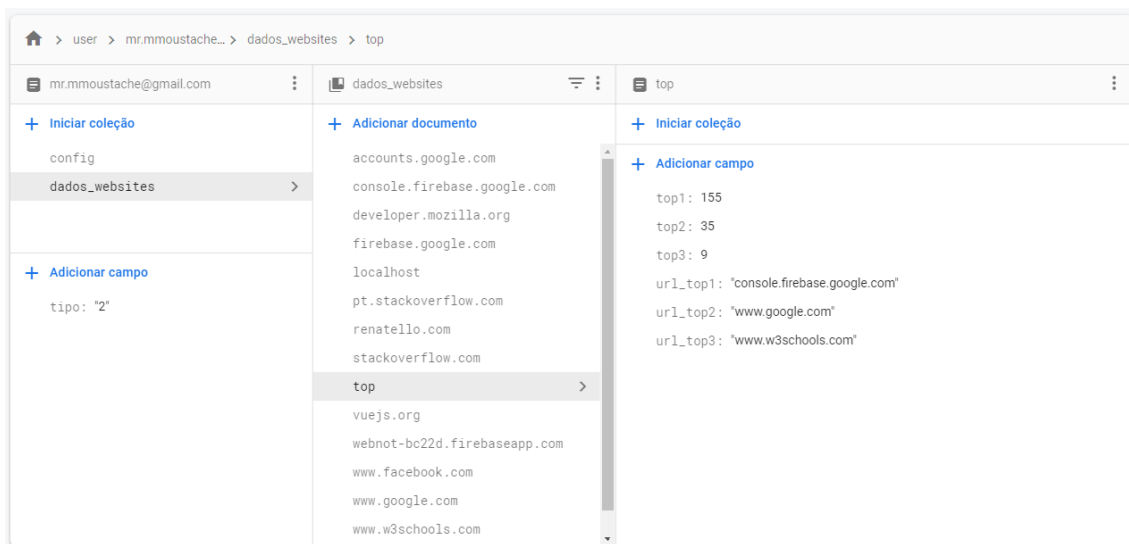


Figura 22 - Firestore (Top3)

6.1.4. Registo das notificações

No registo de notificações, ao iniciar sessão como administrador será apresentado 4 painéis de criação de notificações (Notificações basic, progress, image e list)

A notificação basic é a notificação mais habitual, uma notificação em que são apresentados o título, mensagem da notificação e o seu ícone.

A notificação progresso é uma notificação que é apresentado as mesmas informações que a notificação basic só que também é apresentada uma barra de progresso de 0-100%.

A notificação imagem é uma notificação semelhante a notificação basic mas em que é possível adicionar uma imagem na notificação.

A notificação list é provavelmente a que mais se diferencia das outras, é uma notificação que em vez de ter um campo de texto, é usada uma lista de até 4 itens e que não é possível personalizar o ícone.

Todas as notificações têm o seu próprio ID não podendo ter 2 notificações o mesmo ID.

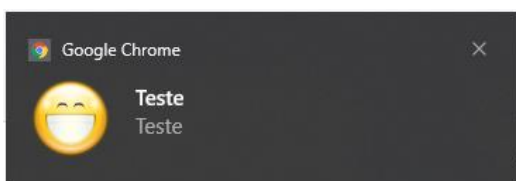


Figura 23 - Notificação basic

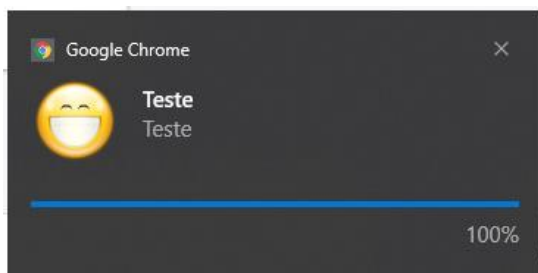


Figura 24 - Notificação progress

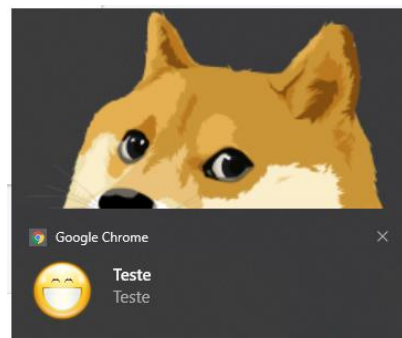


Figura 25 - Notificação image

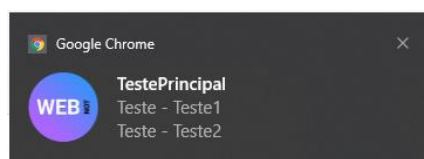


Figura 26 - Notificação list

O registo destas notificações é feito da seguinte forma:

```
93 | //Guarda a notificação do tipo basic na base de dados:
94 | document.querySelector('#save-basic-notification').addEventListener('click', () => {
95 |   if(validateInputs('basic')){
96 |     var id = document.getElementById('basic-id').value;
97 |     db.collection("notificacoesID").doc(id).get().then((doc) => {
98 |       if (doc.exists) {
99 |         createPopup("You can't use that ID!");
100 |       }
101 |       else{
102 |         var imageStorageURL = saveImageOnStorage(id, "basic-icon", "basic", "icon");
103 |
104 |         db.collection("notificacoes").doc("basic").set({
105 |           });
106 |
107 |         db.collection("notificacoes").doc("basic").collection("tipo").doc(document.getElementById('usertype-basic').
108 |         value).get().then((doc) => {
109 |           if (doc.exists) {
110 |             var contagem = doc.data().contaNotificacoes;
111 |             db.collection("notificacoes").doc("basic").collection("tipo").doc(document.getElementById
112 |             ('usertype-basic').value).set({
113 |               contaNotificacoes: contagem + 1
114 |             });
115 |           } else {
116 |             db.collection("notificacoes").doc("basic").collection("tipo").doc(document.getElementById
117 |             ('usertype-basic').value).set({
118 |               contaNotificacoes: 1
119 |             });
120 |           }
121 |         });
122 |
123 |         db.collection("notificacoes").doc("basic").collection("tipo").doc(document.getElementById('usertype-basic').
124 |         value).collection("ID").doc(document.getElementById('basic-id').value).set({
125 |           id: document.getElementById('basic-id').value,
126 |           title: document.getElementById('basic-title').value,
127 |           message: document.getElementById('basic-message').value,
128 |           iconUrl: imageStorageURL
129 |         })
130 |         .then(function() {
131 |           createPopup("Notification saved!");
132 |           console.log("Document successfully written!");
133 |         })
134 |         .catch(function(error) {
135 |           createPopup("Error while saving notification.");
136 |           console.error("Error writing document: ", error);
137 |         });
138 |
139 |         db.collection("notificacoesID").doc(document.getElementById('basic-id').value).set({
140 |           id: document.getElementById('basic-id').value,
141 |           notificacao: "basic",
142 |           tipo: document.getElementById('usertype-basic').value
143 |         })
144 |         .then(function() {
145 |           console.log("Document successfully written!");
146 |         })
147 |         .catch(function(error) {
148 |           createPopup("Error while saving notification.");
149 |           console.error("Error writing document: ", error);
150 |         });
151 |       });
152 |     });
153 |   }
154 | });
```

Figura 27 – main-script.js (Guarda notificação na base de dados)

São feitos dois registos, sendo o primeiro: registo da notificação na coleção “notificacoes” da base de dados por categorias sendo que existe quatro tipos de notificações com diferentes níveis de prioridade ficando registado o seu ID, o título, a mensagem e o URL do ícone ligado ao Firebase Storage.

O segundo registo é feito na coleção “notificacoesID” em que é apenas registado as suas variáveis de identificação que são o seu ID, a categoria da notificação e o seu tipo de prioridade. Este segundo registo é muito importante pois no momento da criação da notificação, é feito uma verificação simples e rápida na coleção “notificacoesID” apenas para verificar se já existe uma notificação com esse ID sem ter de explorar por completo toda a coleção “notificacoes”

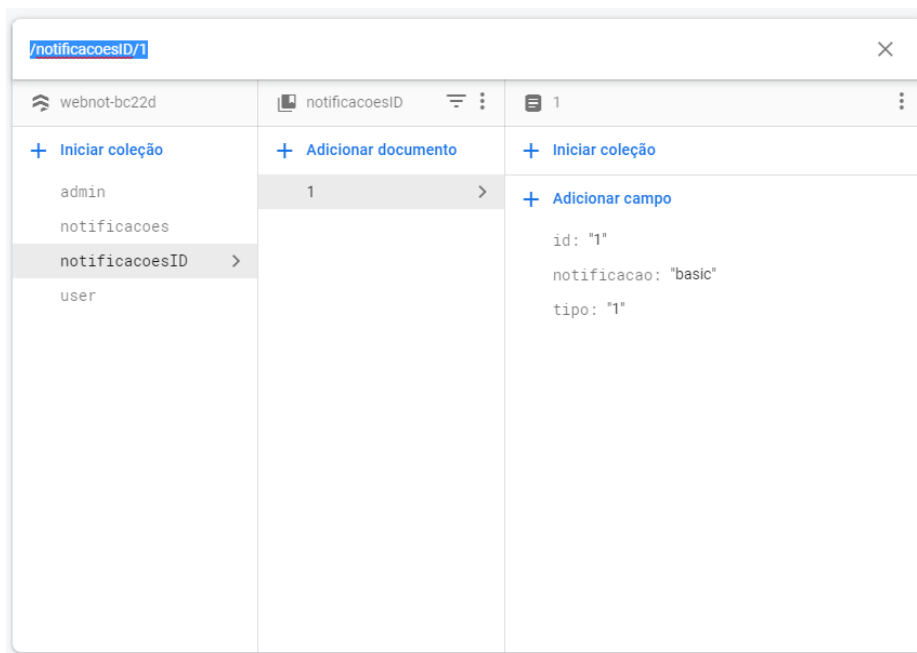


Figura 28 - Firestore (Caminho até ao ID em notificacoesID)

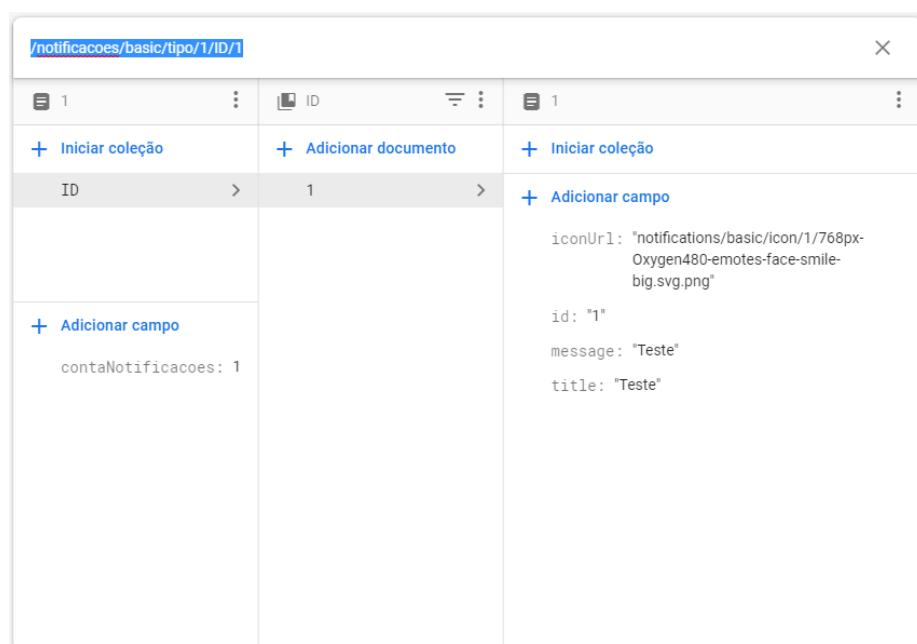


Figura 29 - Firestore (Caminho até ao ID em notificacoes)

6.2. Dashboard do administrador

De forma ao administrador puder aceder as informações dos utilizadores e alterar/eliminar as notificações criadas na extensão Google Chrome, foi criado uma dashboard.

A dashboard foi criada a base de HTML, JavaScript, CSS e Vue.js.

6.2.1. Login

Tal como a extensão, a dashboard também tem o seu login de forma a prevenir que só apenas os administradores tenham acesso aos dados.

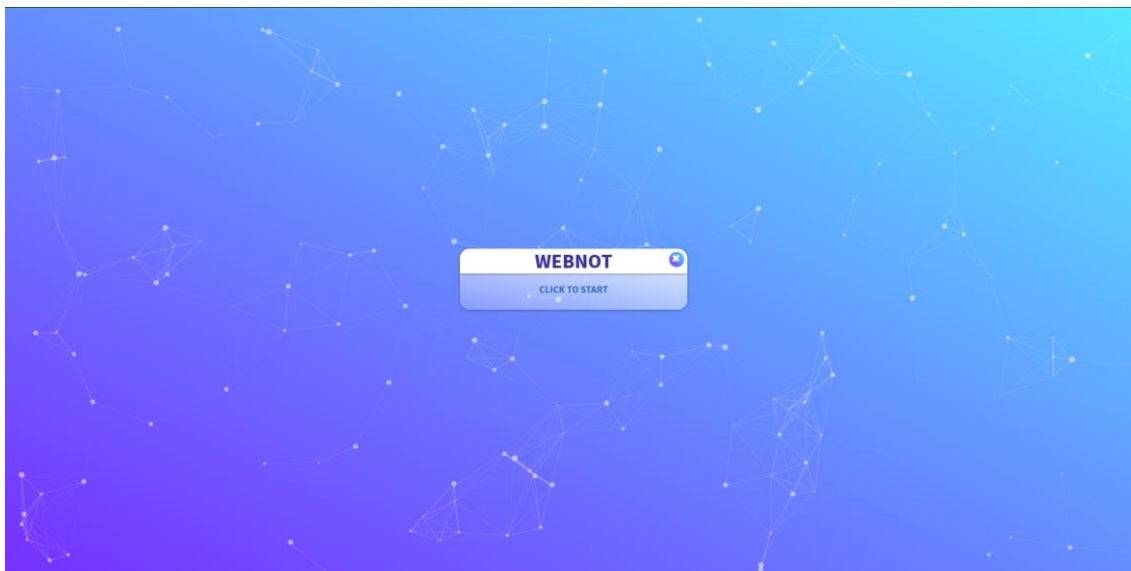


Figura 30 - Dashboard WebNot Start

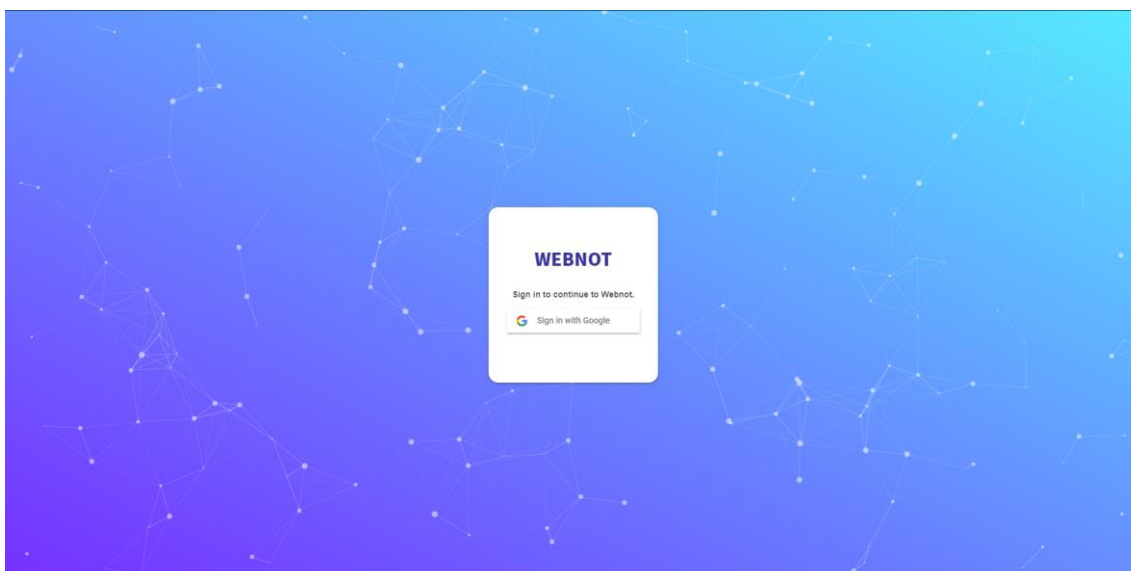


Figura 31 - Dashboard WebNot Login

```

1  const ui = new firebaseui.auth.AuthUI(firebase.auth());
2
3  const uiConfig = {
4    callbacks: {
5      signInSuccessWithAuthResult: function (authResult, redirectUrl) {
6        verificaAdmin();
7      },
8      uiShown: function () {
9        document.getElementById('start').style.display = 'none';
10       document.getElementById('wrapper').style.pointerEvents = 'none';
11     }
12   },
13   signInFlow: 'popup',
14   signInOptions: [
15     {
16       provider: firebase.auth.GoogleAuthProvider.PROVIDER_ID,
17       customParameters: {
18         prompt: 'select_account'
19       }
20     }
21   ],
22 };
23 document.getElementById('start').addEventListener('click', () => {
24   ui.start('#sign_in_options', uiConfig);
25 })
26
27 function verificaAdmin() {
28   var user = firebase.auth().currentUser;
29   if (user != null) {
30     user.providerData.forEach(function (profile) {
31       var docRef = db.collection("admin").doc(profile.email);
32       docRef.get().then((doc) => {
33         if (doc.exists) {
34           localStorage.setItem('emaillogin', profile.email);
35           window.location.replace('http://localhost:3000/html/usuarios.html');
36         } else {
37           window.location.replace('http://localhost:3000/html/nonauthorized.html');
38         }
39       }).catch((error) => {
40         console.log("Erro ao obter o documento:", error);
41       });
42     });
43   }
44 }
45

```

Figura 32 - popup-script.js (Autenticador de login e verificaAdmin())

O login da dashboard é muito semelhante ao login da extensão mudando apenas alguns aspetos, como o tamanho da janela (Tamanho de um separador do browser) e na sua programação foi removido o `verificaUtilizador()` e o email de login é guardado localmente de forma ser verificado se o mesmo pode ter acesso as páginas após o login ou se tentar aceder as páginas sem passar pelo login.

Caso a pessoa que esteja a tentar iniciar sessão não seja administrador será reencaminhada para a página de aviso, indicando que apenas os administradores podem ter acesso aquelas páginas.

6.2.2. Utilizadores

Ao iniciar sessão, o administrador será reencaminhado para a página de utilizadores. Nesta página é apresentado todos os utilizadores que tenham utilizado a extensão.

Após iniciar sessão, o administrador terá sempre acesso aos botões da sidebar “Utilizadores”, “Notificações” e o botão de “Logout” que aparecerá após clicar no ícone do administrador.

ID	Tipo
diegobriceno13@gmail.com	1
mr.moustache@gmail.com	2
sandra.olim@iti.arditi.pt	1

Figura 33 - Utilizadores

Ao clicar em um dos utilizadores será apresentado todos os detalhes do utilizador registados na base de dados, como, o Top 3 páginas mais interagidas, botão “Detalhes” para aceder aos detalhes das páginas acedidas e a capacidade de alterar tipo de prioridade do utilizador.

mr.mmoustache@gmail.com

TOP 3	
#1	console.firebase.google.com 155
#2	www.google.com 35
#3	www.w3schools.com 9

Páginas acedidas

Detalhes

Permissões

Tipo: 2

Guardar

Figura 34- Detalhes de utilizadores

Ao clicar no botão “Detalhes”, será direcionado para as páginas acedidas no qual é possível navegar até aceder aos detalhes da página que foi selecionada na data selecionada.

Websites acedidas

- accounts.google.com
- console.firebase.google.com
- developer.mozilla.org
- firebase.google.com
- localhost
- pt.stackoverflow.com

Figura 35 - Páginas acedidas

Datas em que a página foi acedida

2021-07-02 20:22:48

Voltar

Figura 36 - Datas em que a página foi acedida

Nesta página será apresentado a data do qual foi acedida a página, o seu URL completo e tempo total com a página aberta

accounts.google.com - 2021-07-02 20:22:48

Data:

2021-07-02 20:22:48

Website:

https://accounts.google.com/o/oauth2/auth/oauthchooseaccount?response_type=code&client_id=348923196506-

Tempo total com a página aberta:

657

Voltar

Figura 837 - Detalhes da página anteriormente selecionado na data selecionada

Está página foi desenvolvida com o uso do Vue.js sendo possível navegar na página (alterar o seu conteúdo) numa página única.

Para isso foi utilizado a variável *tiraTabela* para ir alterando o conteúdo apresentado na página.

```
110 <!-- Begin Page Content -->
111 <div class="container-fluid">
112
113   <!-- Page Heading -->
114   <div class="card shadow mb-4" v-if="tiraTabela == 0" > <!--Pagina contendo os utilizadores
115     existentes na BD-->
116     <div class="card-header py-3">
117       <h6 class="m-0 font-weight-bold text-primary">Utilizadores</h6>
118     </div>
119     <div class="card-body">
120       <div class="table-responsive">
121         <table class="table table-bordered" id="dataTable" width="100%" cellspacing="0">
122           <thead>
123             <tr>
124               <th>ID</th>
125               <th>Tipo</th>
126             </tr>
127           </thead>
128           <tbody v-for="(user, index) in users" :key="index">
129             <tr>
130               <td @click="atualizaPagina(user.id)"> {{user.id}}</td>
131               <td> {{user.type}}</td>
132             </tr>
133           </tbody>
134         </table>
135       </div>
136     </div>
137   </div>
```

Figura 38 - Exemplo do uso da variável *tiraTabela*

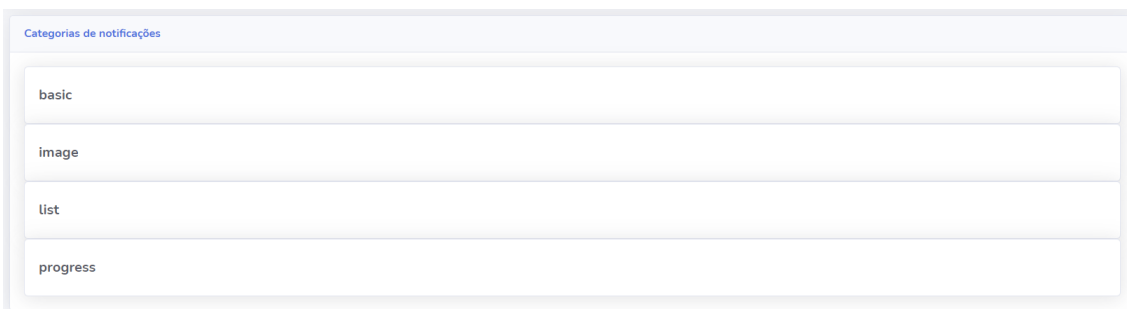
Ao montar tanto a página “utilizadores” como a página “notificações”, é sempre verificado se a variável *emailLogin* do *localStorage* contendo o email do administrador tem permissões para aceder a página, caso não tenha é reencaminhado para a página de aviso anteriormente referida.

```
348 mounted(){ // onload da página, faz a verificação do email com a BD para verificar se é admin, se não
349   for é encaminhado para a página nonauthorized, se for é mostrada a tabela da notificações
350   var emailLogin = localStorage.getItem('emaillogin');
351
352   var docRef = db.collection("admin").doc(emailLogin)
353   docRef.get().then((doc) => {
354     if (!doc.exists) {
355       window.location.replace('nonauthorized.html');
356     } else {
357       db.collection("user").get().then((querySnapshot) => {
358         querySnapshot.forEach((doc) => {
359           let data = doc.data();
360           this.users.push({id: doc.id, type: doc.data().tipo});
361         });
362       });
363     }
364   }).catch((error) => {
365     window.location.replace('nonauthorized.html');
366   });
367 }
```

Figura 939 - *mounted()* de *utilizadores.html*

6.2.3. Notificações

Ao aceder a página de notificações através do botão “Notificações” da sidebar, será apresentado as 4 categorias de notificações (“Basic”, “Progress”, “Image”, “List”).



Categories de notificações

basic

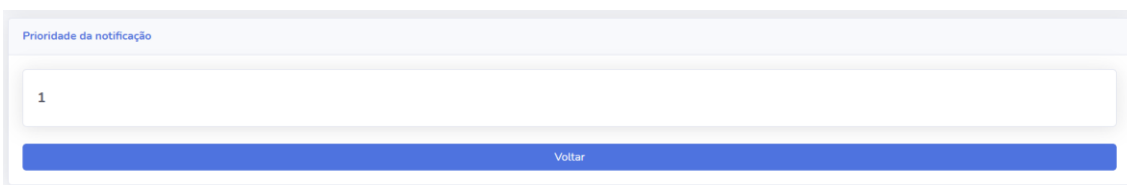
image

list

progress

Figura 40 - Notificações

Nesta página, a navegação é feita da mesma forma que na página de utilizadores, é utilizado o Vue.js para uma navegação numa página única.

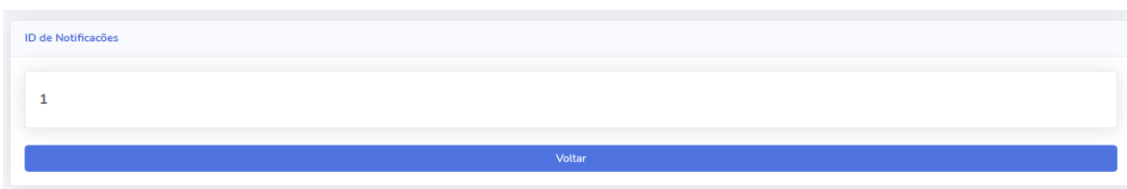


Prioridade da notificação

1

Voltar

Figura 41 - Prioridade de notificação

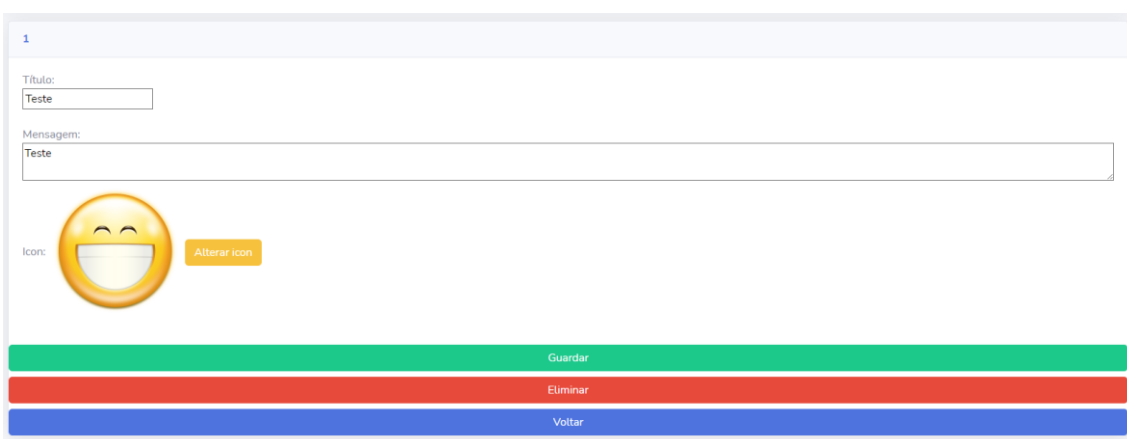


ID de Notificações

1

Voltar


Figura 42 - ID de Notificações



1

Título:
Teste

Mensagem:
Teste

Icon:
 Alterar icon

Guardar

Eliminar

Voltar

Figura 43 - Detalhes da notificação

Ao chegar a página de detalhes de notificação, é apresentado os dados guardados na base de dados correspondente a notificação selecionada. Nesta página o administrador pode alterar os dados da notificação e clicar “Guardar” ou eliminar a notificação, clicando no botão “Eliminar”

6.2.3.1. Editar notificação

Ao editar a notificação, se o administrador estiver em dúvida se pretende ou não alterar o ícone, é possível pré-visualizar sem que o administrador tenha de clicar no botão “Guardar”. Para isto basta clicar no botão “Alterar icon” e seleccionar a ícone desejado.

Caso o administrador se goste do ícone seleccionada, basta clicar no botão “Voltar” e voltar a clicar no ID da notificação que pretende editar.

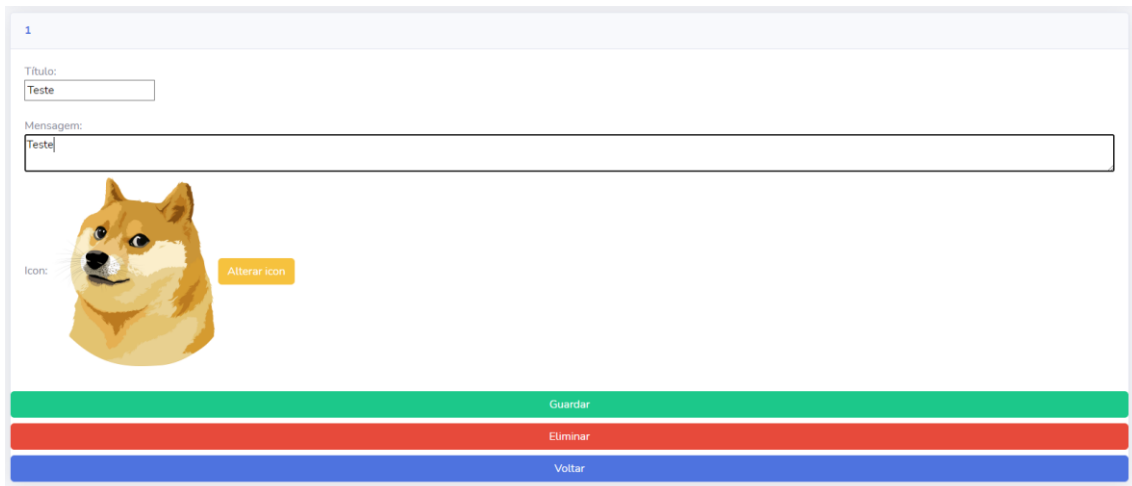


Figura 44 - Pré-visualização do ícone

```
388 //Ao editar uma notificação "basic", "progress" ou "image", é dada uma preview da imagem que foi
389 carregada pelo utilizador
390 previewImage(idElement,element){
391     var input = document.getElementById(idElement);
392     if (input.files && input.files[0]) {
393         console.log(input);
394         var reader = new FileReader();
395
396         reader.onload = function(e) {
397             document.getElementById(element).src=e.target.result;
398         }
399         reader.readAsDataURL(input.files[0]);
400     }
401 },
```

Figura 45 - Função previewImage() de notificacoes.html

Ao clicar no botão “Guardar”, é necessário ter em atenção os campos preenchidos, pois não é possível deixar nenhum dos campos vazios, ao deixar um dos campos vazios é dada uma mensagem de aviso.

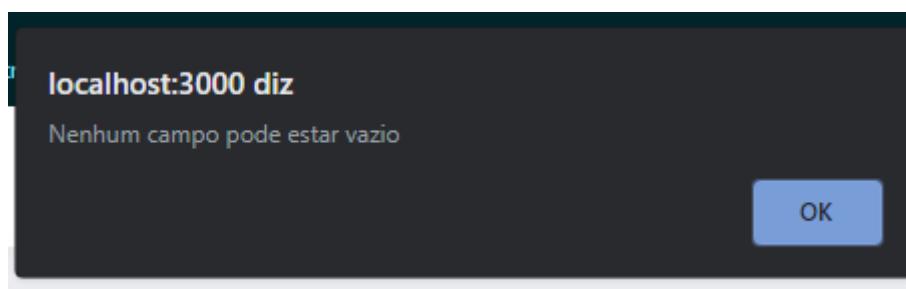


Figura 46 - Mensagem de aviso

No caso de todos os campos estarem preenchidos, ao clicar no botão “Guardar”, é verificado qual a categoria da notificação e se alguma imagem foi carregada para alterar a imagem atual do ícone e/ou da imagem (Notificação “image”), caso não tenha sido carregada nenhuma imagem será feito o update dos dados da notificação, caso tenha sido carregada, é feita a alteração da imagem e o update dos dados.

```

414 //Após editar a notificação "basic", "progress" ou "image", ao clicar no botão "Guardar", é feito o
415 update dos dados alterados na BD e no Storage (Caso seja feitas alterações no icon e/ou imagem)
416 guardar(){
417     if(this.titleNotificacao == "" || this.messageNotificacao == ""){
418         alert("Nenhum campo pode estar vazio");
419     }
420     else{
421         if(this.categoria == "basic"){
422             var fileUploaded = document.getElementById('basic-icon').files[0];
423         }
424         if(this.categoria == "progress"){
425             var fileUploaded = document.getElementById('progress-icon').files[0];
426         }
427         if(this.categoria == "image"){
428             var fileUploaded = document.getElementById('image-icon').files[0];
429             var fileUploadedImagem = document.getElementById('image-image').files[0];
430             if(fileUploadedImagem != null){
431                 var storageRefNovo = firebase.storage().ref('notifications/'+this.categoria+'/'+
432                 'image'+ '/' +this.IDnotificacao+'/' +fileUploadedImagem.name);
433                 storageRefNovo.put(fileUploadedImagem);
434                 // Delete the file
435                 storageRef.child(this.imageUrlNotificacao).delete().then(function() {
436                     console.log('File deleted successfully')
437                 }).catch(function(error) {
438                     // Uh-oh, an error occurred!
439                 });
440                 db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).
441                 collection("ID").doc(this.IDnotificacao).update({
442                     imageUrl: storageRefNovo.fullPath
443                 });
444                 this.imageUrlNotificacao = storageRefNovo.fullPath;
445             }
446             if(fileUploaded != null){
447                 var storageRefNovo = firebase.storage().ref('notifications/'+this.categoria+'/' +this.IDnotificacao+'/' +fileUploaded.name);
448                 storageRefNovo.put(fileUploaded);
449                 // Delete the file
450                 storageRef.child(this.iconUrlNotificacao).delete().then(function() {
451                     console.log('File deleted successfully')
452                 }).catch(function(error) {
453                     // Uh-oh, an error occurred!
454                 });
455                 db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).
456                 collection("ID").doc(this.IDnotificacao).update({
457                     iconUrl: storageRefNovo.fullPath
458                 });
459                 this.iconUrlNotificacao = storageRefNovo.fullPath;
460             }
461             db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).
462             collection("ID").doc(this.IDnotificacao).update({
463                 title: this.titleNotificacao,
464                 message: this.messageNotificacao
465             });
466             alert("Notificacao atualizada");
467         }
468     },

```

Figura 47 - Função guardar() de notificacoes.html

6.2.3.2. Eliminar notificação


No caso de o administrador entender que já não é necessário o uso da notificação, é possível eliminá-la, clicando no botão “Eliminar”.

Ao clicar no botão eliminar é executado o seguinte código:

```
567 //Visualizando os detalhes da notificação, ao clicar no botão "Eliminar", é feita eliminação dos
568 dados da notificação na BD e o icon (e imagem, caso seja uma notificacao "image") do storage
569 eliminaNotificacao(){
570     db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).collection
571     ("ID").doc(this.IDnotificacao).delete().then(() => {
572         console.log("Document successfully deleted!");
573     }).catch((error) => {
574         console.error("Error removing document: ", error);
575     });
576     db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).get().then(
577     (doc) => {
578         var contagem = doc.data().contaNotificacoes;
579         db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).update({
580             contaNotificacoes: contagem - 1
581         }).catch((error) => {
582             console.error("Error removing document: ", error);
583         });
584     });
585     db.collection("notificacoesID").doc(this.IDnotificacao).delete().then(() => {
586         console.log("Document successfully deleted!");
587     }).catch((error) => {
588         console.error("Error removing document: ", error);
589     });
590     if(this.categoria == "basic" || this.categoria == "progress" || this.categoria == "image")
591     storageRef.child(this.iconUrlNotificacao).delete().then(function() {
592         console.log('File deleted successfully')
593     }).catch(function(error) {
594         // Uh-oh, an error occurred!
595     });
596     if(this.categoria == "image") {
597         storageRef.child(this.imageUrlNotificacao).delete().then(function() {
598             console.log('File deleted successfully')
599         }).catch(function(error) {
600             // Uh-oh, an error occurred!
601         });
602     }
603     db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).get().then(
604     (doc) => {
605         var contagem = doc.data().contaNotificacoes;
606         if(contagem == 1){
607             db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).
608             delete().then(() => {
609                 console.log("Document successfully deleted!");
610             }).catch((error) => {
611                 console.error("Error removing document: ", error);
612             });
613             this.notificacoes = [];
614             this.tipos = [];
615             this.tiraTabela = '1';
616         } else {
617             this.notificacoes = [];
618             this.tiraTabela = '3';
619             db.collection("notificacoes").doc(this.categoria).collection("tipo").doc(this.tipo).
620             collection("ID").get().then((querySnapshot) => {
621                 querySnapshot.forEach((doc) => {
622                     this.notificacoes.push({id: doc.id});
623                 });
624             });
625         }
626     });
627 }
```

Figura 48 - Função `eliminaNotificacao()` de `notificacoes.html`

A função `eliminaNotificacao()`, elimina os dados na coleção “notificacoes” e “notificacoesID” da base de dados referentes a notificação selecionada, elimina a imagem/ns (caso seja uma notificação “image”, tem ícone e imagem) ligada ao ID da notificação no Firebase Storage e após



isso, o administrador é reencaminhado para a tabela de “IDs de Notificações” se ainda existir mais alguma notificação com o mesmo tipo de prioridade que a notificação eliminada, senão, é reencaminhado para o menu principal das notificações, onde é apresentado as 4 categorias de notificações.

7. Conclusão

Ao longo do desenvolvimento deste trabalho foram encontrados vários problemas, sendo o primeiro problema a escolha da base de dados, no processo inicial a base de dados discutida teria sido o “MySQL” pois era a base de dados do qual tínhamos mais conhecimento adquirido ao longo do percurso académico mas não foi possível devido a várias complicações com scripts que não poderiam ser executados em extensões Chrome por diversas restrições colocadas pela Google, após uma nova discussão, foi escolhido a base de dados “Firestore” da Firebase, esta escolha foi feita tendo em conta as diversas tecnologias que o Firebase fornece, desde a própria base de dados, autenticação de login e armazenamento de ficheiros.

O segundo problema ocorreu nas fases finais do projeto, no desenvolvimento da dashboard do administrador, como forma de o administrador poder gerir os dados dos utilizadores e as notificações. Neste caso foi a dificuldade de enviar informação de uma página para outra, para este problema, a solução escolhida foi o uso do “Vue.js”. Esta solução ajudou imenso, pois assim, foi possível o desenvolvimento de uma interface de página única, facilitando imenso no envio de dados sem que fosse necessário alterar a página.

Em suma, o problema proposto foi resolvido, pois a extensão criada permite fazer exatamente o que tinha sido proposto como solução.

A extensão exibe o tempo total utilizando o browser permitindo lembrar o utilizador sobre fazer uma pausa de forma a poder aumentar o desempenho nas suas tarefas quando voltar, visualizar há quanto tempo está a utilizar um determinado website e receber determinadas notificações dependendo do tipo de prioridade do utilizador.

Permite com que o administrador crie diversas notificações de várias categorias (“Basic”, “Progress”, “Image” e “List”) para diversos tipos de prioridade, guardando-as na base de dados que posteriormente pode acessar a partir da dashboard para que possa fazer alterações ou eliminá-las, ainda na dashboard é possível verificar os dados registados dos utilizadores que utilizaram a extensão.

8. Referências

- [1] S. Park *et al.*, «Effects of display curvature and task duration on proofreading performance, visual discomfort, visual fatigue, mental workload, and user satisfaction», *Applied Ergonomics*, vol. 78, pp. 26–36, Jul. 2019, doi: 10.1016/j.apergo.2019.01.014.
- [2] D. Weber, S. Mayer, A. Voit, R. Ventura Fierro, e N. Henze, «Design Guidelines for Notifications on Smart TVs», em *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, Chicago Illinois USA, Jun. 2016, pp. 13–24. doi: 10.1145/2932206.2932212.
- [3] D. Weber, A. Voit, e N. Henze, «Notification Log: An Open-Source Framework for Notification Research on Mobile Devices», em *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, Singapore Singapore, Out. 2018, pp. 1271–1278. doi: 10.1145/3267305.3274118.
- [4] «What is push notification? - Definition from WhatIs.com», *SearchMobileComputing*. <https://searchmobilecomputing.techtarget.com/definition/push-notification>.
- [5] «What is pull notification? - Definition from WhatIs.com», *SearchMobileComputing*. <https://searchmobilecomputing.techtarget.com/definition/pull-notification>.
- [6] «Cloud Firestore | Store and sync app data at global scale», *Firebase*. <https://firebase.google.com/products/firestore?hl=pt>.
- [7] «Cloud Firestore», *Firebase*. <https://firebase.google.com/docs/firestore?hl=pt>.
- [8] «Firebase Authentication | Simple, free multi-platform sign-in», *Firebase*. <https://firebase.google.com/products/auth?hl=pt>.
- [9] «Cloud Storage para Firebase», *Firebase*. <https://firebase.google.com/docs/storage?hl=pt>.
- [10] «Extensions», *Chrome Developers*. <https://developer.chrome.com/docs/extensions/>.
- [11] «Documentation for Visual Studio Code». <https://code.visualstudio.com/docs>.
- [12] «HTML: Linguagem de Marcação de Hipertexto | MDN». <https://developer.mozilla.org/pt-BR/docs/Web/HTML>.
- [13] «JavaScript - Aprendendo desenvolvimento web | MDN». <https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript>.
- [14] «Vue.js», *Wikipédia, a enciclopédia livre*. Dez. 30, 2020. Disponível em: <https://pt.wikipedia.org/w/index.php?title=Vue.js&oldid=60127884>