

**INSTITUTO POLITÉCNICO DE BEJA**  
**Escola Superior de Tecnologia e Gestão**  
**Licenciatura em Engenharia Informática**



**Tecnologias Web e Ambientes Móveis**  
**Projeto – Fase de Implementação**  
**Ano Letivo 2021/2022 / 2º Semestre / 2º Ano**

Elaborado por:

Fábio Gonçalves nº 17646

Docente:

Luís Carlos Bruno

08/07/2022

# Índice

1. Introdução.....	3
2. Decisões Globais de Implementação .....	4
2.1. Tecnologias de desenvolvimento utilizadas .....	4
2.2. Definições usadas para assegurar a responsividade para diferentes dimensões e resoluções dos dispositivos.....	4
2.3. Decisões gerais aplicadas em termos de acessibilidade para pessoas com necessidades especiais .....	5
3. Decisões de Implementação Específicas .....	6
3.1. Redesenho de Interfaces .....	6
3.2. Criação de um projeto na consola do Google API.....	8
3.3. Adicionar API do Gmail ao projeto .....	8
3.4. Credenciais e autenticação com OAuth 2.0 .....	8
3.5. Tela de permissões OAuth.....	9
3.6. Componente App.js .....	10
3.7. Homepage .....	11
3.8. Enviar Email / Rascunho.....	14
3.9. Tarefa 1 – Enviar Email .....	14
3.10. Tarefa 2 – Criar Rascunho .....	15
3.11. Tarefa Extra – Visualizar / Utilizar Rascunhos.....	16
3.12. Tarefa Extra – Consultar Emails Enviados .....	19
3.13. Expirar da sessão.....	21
4. Avaliação das Interfaces / Testes com Utilizadores .....	22
4.1. Guião.....	22
5. Conclusões Finais.....	23

## Índice de figuras

Figura 1 - Informar o utilizador da funcionalidade do botão .....	5
Figura 2 - Informar ao utilizador que pode clicar no assunto e assim obter mais informações.5	
Figura 3 - Informar ao utilizador que pode clicar no assunto e obter o rascunho.....5	
Figura 4 - Atributo alternativo de imagem (alt).....5	
Figura 5 - Interface "Enviar Email" previamente desenvolvida .....	6
Figura 6 - Interface "Rascunhos" previamente desenvolvida .....	7
Figura 7 - Interface de "Enviar Email / Rascunho" atual.....7	
Figura 8 - Ativação da Gmail API no projeto .....	8
Figura 9 - Credenciais (Chave de API) .....	8
Figura 10 - Credenciais (ID do cliente OAuth 2.0).....8	
Figura 11 – Atribuição de Escopos .....	9
Figura 12 - Atribuição dos utilizadores.....10	
Figura 13 - Componente App.js.....10	
Figura 14 - Página Inicial.....11	
Figura 15 - Pop-up de login com conta gmail (conceder as devidas permissões) .....	11
Figura 16 - Conexão do website ao projeto no Google console.....12	
Figura 17 - Autenticação de utilizador através da token.....13	
Figura 18 - Interface "Enviar Email / Rascunho" .....	14
Figura 19 - Função para enviar email .....	15
Figura 20 - Popup (Email enviado com sucesso).....15	
Figura 21 - Função para criação de um rascunho .....	16
Figura 22 - Popup (Rascunho guardado com sucesso) .....	16
Figura 23 - Executar função "showDrafts()" após interface ser inicializada .....	17
Figura 24 - Função para mostrar rascunhos criados .....	17
Figura 25 - Criação da tabela rascunhos e click (jQuery) .....	18
Figura 26 - Funções getHeader() e getBody() .....	18
Figura 27 - message.payload.....19	
Figura 28 - Executar função "showSentEmails()" após interface ser inicializada.....19	
Figura 29 - Função para mostrar emails enviados .....	19
Figura 30 - Criação da tabela de emails enviados e popup (jQuery).....20	
Figura 31 - Popup de consultar email enviado.....20	
Figura 32 - Popup de confirmação após tentar atualizar ou fechar a janela do browser .....	21
Figura 33 - Interface de nenhuma sessão iniciada .....	21

## Índice de Tabelas

Tabela 1 - API Requests .....	3
Tabela 2 - Tecnologias e as suas respetivas versões utilizadas.....4	
Tabela 3 - Resultados dos testes com utilizadores .....	22

# 1. Introdução

Na segunda fase deste projeto foram então implementados os aspetos descritos previamente na fase um de análise, em que foram implementadas as seguintes duas tarefas:

- Enviar emails;
- Enviar Rascunhos.

Foram também implementadas outras duas tarefas extra que complementam as anteriores, essas tarefas foram:

- Visualizar emails enviados;
- Utilizar rascunhos previamente criados.

As *requests* que foram utilizadas para o uso da *API* e o cumprimento de todas as tarefas descritas acima foram as seguintes:

Método	Resquest	Descrição
POST	https://gmail.googleapis.com/upload/gmail/v1/users/{userId}/messages/send	Enviar um email.
POST	https://gmail.googleapis.com/upload/gmail/v1/users/{userId}/drafts/send	Enviar um rascunho.
GET	https://gmail.googleapis.com/gmail/v1/users/{userId}/messages	Obter os ids das mensagens enviadas através do parâmetro "SENT"
GET	https://gmail.googleapis.com/gmail/v1/users/{userId}/messages/{id}	Obtém o contexto das mensagens através do id obtido de "SENT"
GET	https://gmail.googleapis.com/gmail/v1/users/{userId}/messages	Obter os ids dos rascunhos criados através do parâmetro "DRAFT"
GET	https://gmail.googleapis.com/gmail/v1/users/{userId}/messages/{id}	Obtém o contexto dos rascunhos através do id obtido de "DRAFT"

**Tabela 1 - API Resquests**

Ao longo deste relatório será então demonstrado todo o processo realizado no desenvolvimento da implementação, tanto a parte técnica como na prática o uso das mesmas, em que serão descritos os processos para a realização das tarefas e o funcionamento das mesmas seguidos por imagens e uma breve explicação das mesmas.

## 2. Decisões Globais de Implementação

Neste tópico serão descritas todas as decisões de forma simples e detalhada das tecnologias, diferentes tipos de implementação tomadas nas escolhas em termos de funcionalidades como acessibilidade no que toca ao sistema.

### 2.1. Tecnologias de desenvolvimento utilizadas

Descritos os diferentes tipos de tecnologias utilizadas para o desenvolvimento do projeto e as suas respetivas versões.

Tecnologia	Versão
<a href="#">PhpStorm</a>	213.5744.279
<a href="#">NodeJS</a>	16.15.0
<a href="#">React</a>	18.2.0

Tabela 2 - Tecnologias e as suas respetivas versões utilizadas


### 2.2. Definições usadas para assegurar a responsividade para diferentes dimensões e resoluções dos dispositivos

No que toca ao assegurar a responsividade do website em diferentes dimensões e resoluções de diferentes dispositivos em que o mesmo possa ser executado foram aplicadas as devidas mudanças no código CSS que grande parte dos tamanhos dos diferentes estilos e componentes utilizam o tamanho em medida percentual (%) para assim assegurar que se adaptam a diferentes tipos de tamanho visto que essa medida varia consoante a resolução definida do dispositivo que se encontra de momento a visualizar o website.

Foram também aplicadas as mesmas medidas para tabelas e *containers* em que o seu conteúdo se ajusta dinamicamente à resolução do utilizador, tudo isto foi implementado no ficheiro de CSS utilizado pelo sistema.

### 2.3. Decisões gerais aplicadas em termos de acessibilidade para pessoas com necessidades especiais

Para facilitar a utilização do website a utilizadores com menos capacidades de compreensão ou percepção são indicados pequenos textos seguidos do *icon* de informação

(  ) para informar o utilizar da ação que pode realizar naquele momento caso o mesmo não o tenha apercebido e também assim ajudar a “guiá-lo” para cumprir com as suas funcionalidades, nas figuras abaixo podemos confirmar as mesmas.

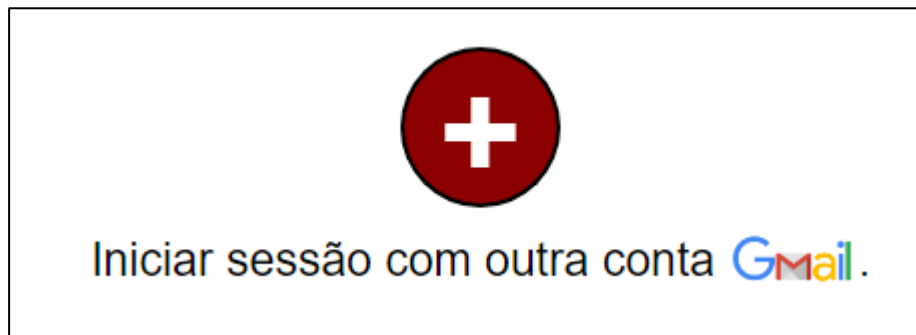


Figura 1 - Informar o utilizador da funcionalidade do botão

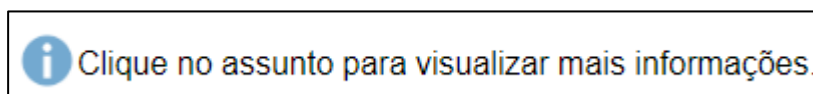


Figura 2 - Informar ao utilizador que pode clicar no assunto e assim obter mais informações

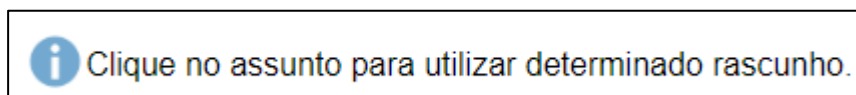


Figura 3 - Informar ao utilizador que pode clicar no assunto e obter o rascunho

Outra acessibilidade que o website tem é o facto de as imagens possuírem o campo *alt* em que apesar de esse campo servir de *placeholder* para se a imagem não for carregada serve também para utilizadores que tenham problemas visuais ou até mesmo não terem visão nenhuma e assim ao terem o *text-to-speech* ativo o browser irá imitar o áudio da palavra que foi escrita no campo *alt* da imagem como podemos verificar na figura abaixo.

```
<img src={email} className="size" alt="Imagem de Email"/>
```

Figura 4 - Atributo alternativo de imagem (*alt*)

### 3. Decisões de Implementação Específicas

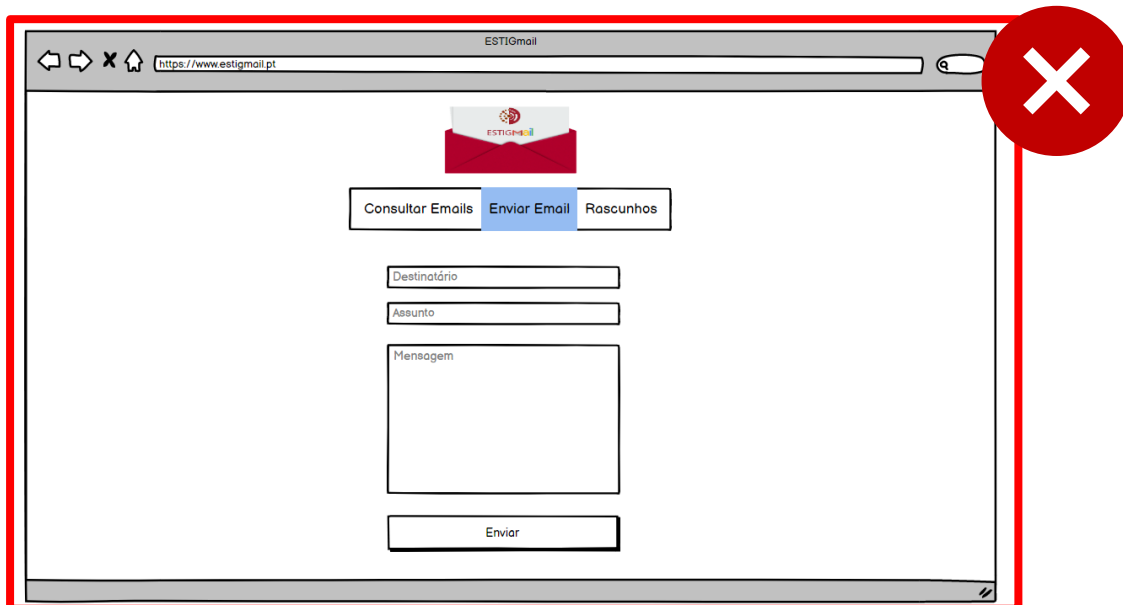
Neste tópico serão abordadas todas as principais decisões no que toca à implementação para a devida funcionalidade que suporta cada um dos elementos que foi implementado no sistema.

#### 3.1. Redesenho de Interfaces

Após começar a implementar o sistema deparei-me com o facto de estar a implementar a interface de “Enviar Email” e “Criar Rascunho” em páginas do website distintas, o que poderia confundir o utilizador e o que também me foi sugerido por um amigo a quem pedi recomendações no que toca ao desenho das interfaces.

Visto que as interfaces se podem complementar uma à outra, como por exemplo, a criação de um rascunho pode resultar no envio desse rascunho como mensagem e a criação de uma mensagem pode resultar em guardar essa mensagem como rascunho, fez mais sentido realizar então a implementação de ambas as tarefas na mesma interface, para assim tornar também mais fácil e organizada a experiência do utilizador.

Abaixo podemos visualizar o esboço de média fidelidade realizado previamente e o novo com as alterações estrategicamente implementadas.



**Figura 5 - Interface "Enviar Email" previamente desenvolvida**

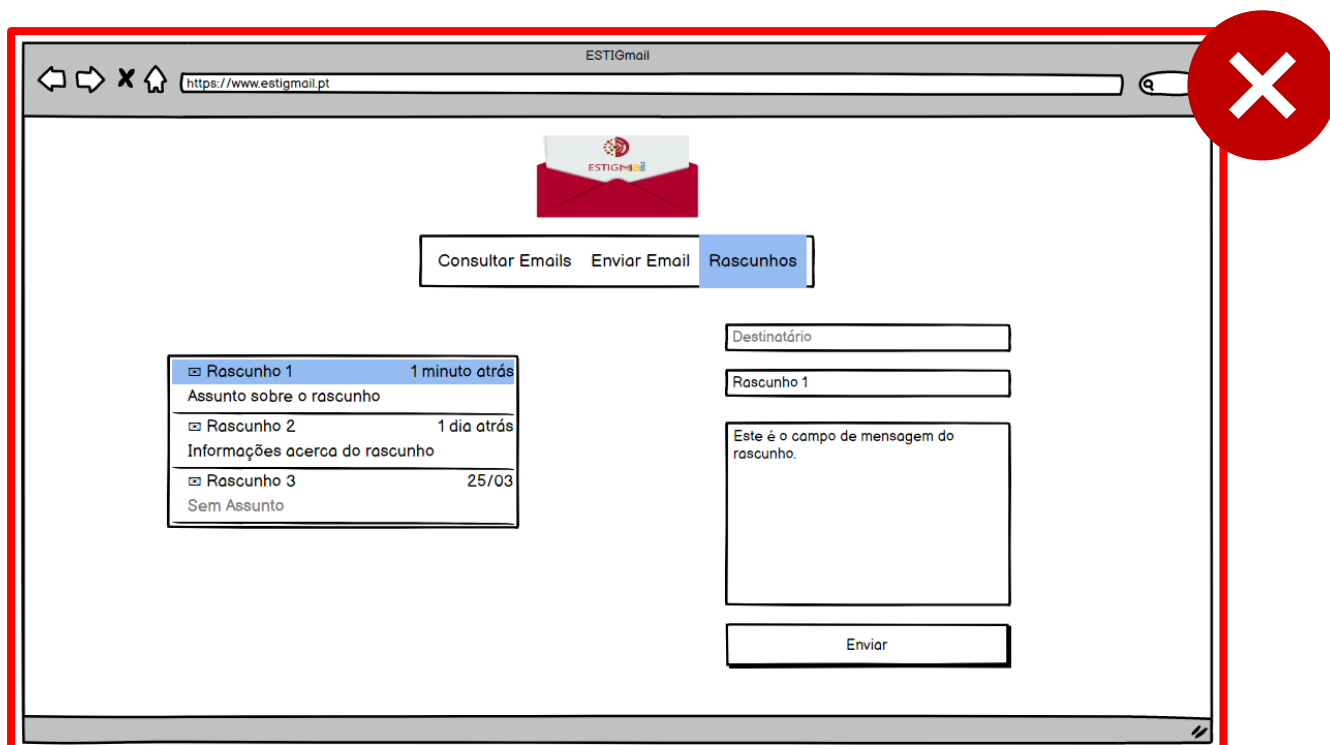


Figura 6 - Interface "Rascunhos" previamente desenvolvida

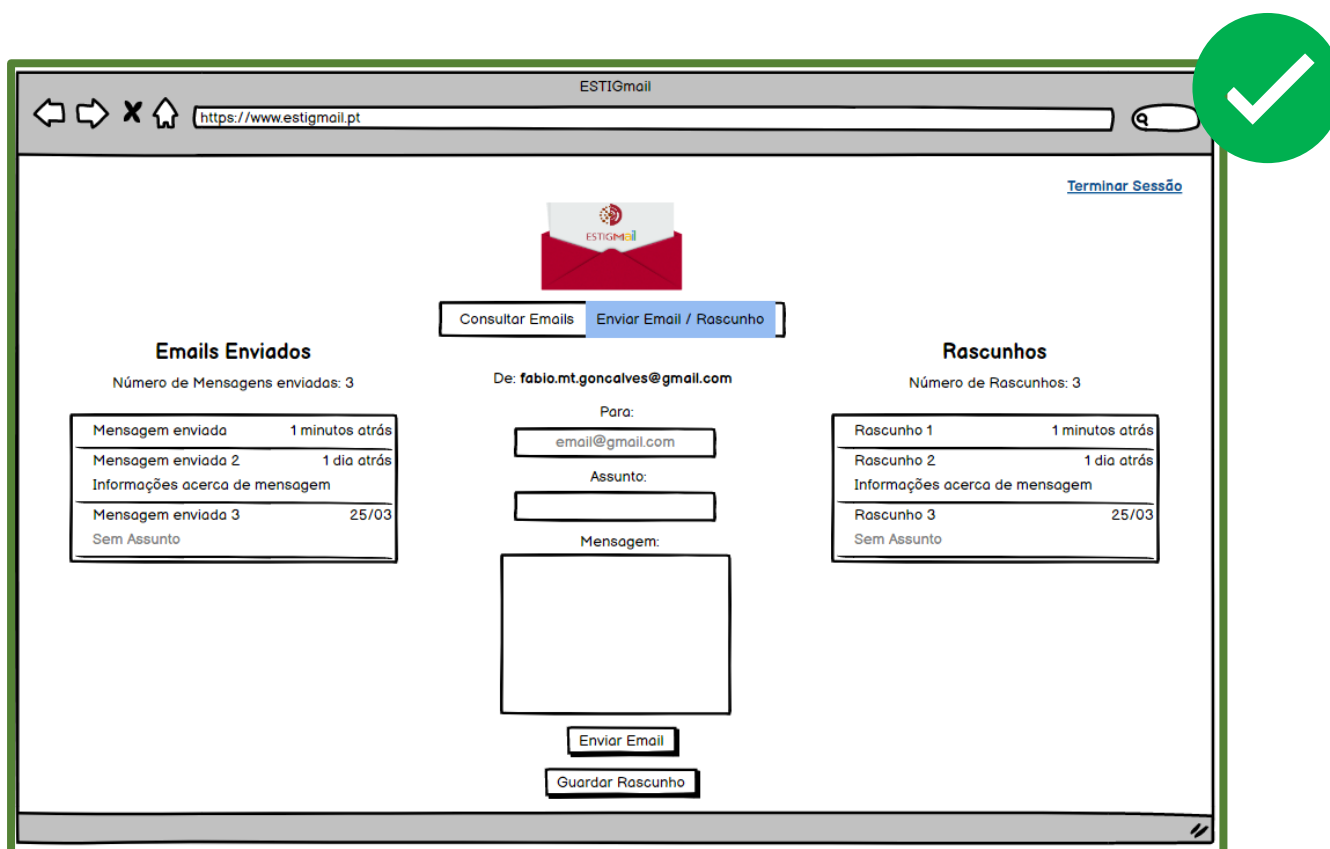


Figura 7 - Interface de "Enviar Email / Rascunho" atual



### 3.2. Criação de um projeto na consola do Google API

Primeiramente antes de começar a implementação do código que irá comunicar com a *API* foram precisos efetuar alguns passos em primeiro lugar para que tudo possa ser efetuado com sucesso, e é isso que será abordado neste tópico.

### 3.3. Adicionar API do Gmail ao projeto

Após a criação do projeto na consola do Google API foi habilitada o acesso da *API* do Gmail ao projeto através da biblioteca de *API's* da Google como se pode verificar na figura abaixo.

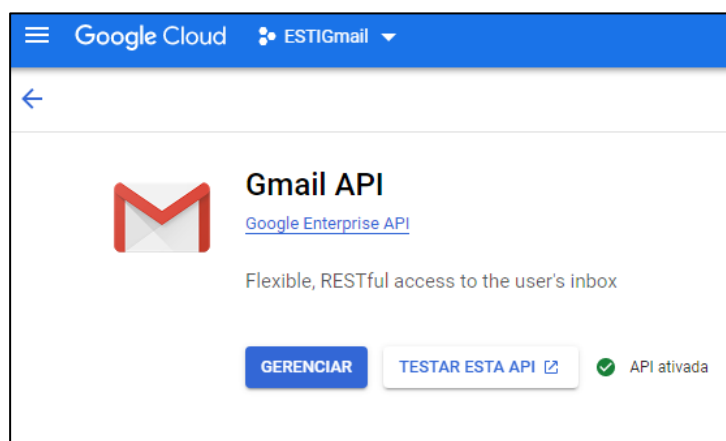


Figura 8 - Ativação da Gmail API no projeto

### 3.4. Credenciais e autenticação com OAuth 2.0

Para o uso da *API* do Gmail é necessária autenticação com OAuth 2.0 para serem fornecidas credenciais de acesso que neste caso as credenciais necessárias para poder utilizar a *API* do Gmail.

No que toca às credenciais foi criada uma chave API e uma ID de cliente OAuth 2.0, em que será essa chave e essa ID que irão conectar o nosso sistema criado em PhpStorm ao nosso projeto do Google Cloud que irá então estar conectado com a *API* do Gmail.

Chaves de API		
<input type="checkbox"/>	Nome ↑	Data da criação ↓
<input type="checkbox"/>	Chave de API 3	25 de jun. de 2022
		Restrições
		Nenhum

Figura 9 - Credenciais (Chave de API)

IDs do cliente OAuth 2.0			
<input type="checkbox"/>	ESTIGmail	23 de mai. de 2022	Aplicativo da Web
			436969203574-07mg...

Figura 10 - Credenciais (ID do cliente OAuth 2.0)

### 3.5. Tela de permissões OAuth

Nas permissões da parte do uso de OAuth foram adicionados os emails dos devidos utilizadores a terem acesso às credenciais do respetivo projeto e assim conseguirem fazer requests à API do Gmail por parte do projeto no qual estão associados, que no caso será o “ESTIGmail”.

Ainda na parte de permissões OAuth foram também adicionados os Scopes, que serão as permissões que se forem autorizadas do lado do projeto e implementadas no sistema o utilizador terá então de concordar com as mesmas para que o website tenha acesso a essas permissões para efetuar as tarefas do sistema, neste caso foram adicionados dos scopes relacionados ao Gmail mas na parte de implementação foram apenas usados quatro:

- `'https://mail.google.com/'`
- `'https://www.googleapis.com/auth/gmail.readonly'`
- `'https://www.googleapis.com/auth/gmail.send'`
- `'https://www.googleapis.com/auth/userinfo.profile'`

<input type="checkbox"/>	API ↑	Escopo	Descrição voltada para o usuário
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.addons .current.message .metadata	Ver os metadados da sua mensagem de e-mail quando o complemento estiver sendo executado
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.addons .current.message .readonly	Ver suas mensagens de e-mail quando o complemento estiver sendo executado
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.send	Enviar e-mail no seu nome
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.labels	Ver e editar os marcadores do seu e-mail
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.settings .basic	Ver, editar, criar ou mudar seus filtros e suas configurações de e-mail no Gmail
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.settings .sharing	Gerenciar todas as configurações confidenciais, inclusive quem pode gerenciar seu e-mail
<input checked="" type="checkbox"/>	Gmail API	https://mail.google.com/	Ler, escrever, enviar e excluir permanentemente todos os seus e-mails do Gmail
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.modify	Ler, escrever e enviar e-mails da sua conta do Gmail
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.compose	Gerenciar rascunhos e enviar e-mails
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.addons .current.action .compose	Gerenciar rascunhos e enviar e-mails quando você interagir com o complemento
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.addons .current.message .action	Ver suas mensagens de e-mail quando você interagir com o complemento
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.readonly	Ver suas configurações e mensagens de e-mail
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.metadata	Ver os metadados da mensagem de e-mail, como marcadores e cabeçalhos, mas não o corpo do e-mail
<input checked="" type="checkbox"/>	Gmail API	.../auth/gmail.insert	Adicionar mensagens à caixa de e-mails do Gmail

Figura 11 – Atribuição de Escopos

Usuários de teste	
+ ADD USERS	
<div>Filtro</div> <div>Insira o nome ou o valor da propriedade</div> <div>?</div>	
Informações do usuário	
best.of.football.yt@gmail.com	
fabio.mt.goncalves@gmail.com	
fabolas_wwe@hotmail.com	
portelinhapt@gmail.com	
scmkasd1@gmail.com	
twam17646@gmail.com	
twam20481@gmail.com	

Figura 12 - Atribuição dos utilizadores

### 3.6. Componente App.js

App.js é o componente root que faz a correlação entre todas as páginas que existem no projeto através de Routers e Switches o que assim faz com que seja possível os utilizadores navegarem de página para página.

```
function App() {

  return (
    <Router>
      <Switch>
        <Route path="/" exact component={Home} />
        <>
          <Navbar />
          <Route path="/EnviarEmail" component={EnviarEmail} />
          <Route path="/EnviarRascunho" component={EnviarRascunho} />
        </>
      </Switch>
    </Router>
  );
}

export default App;
```

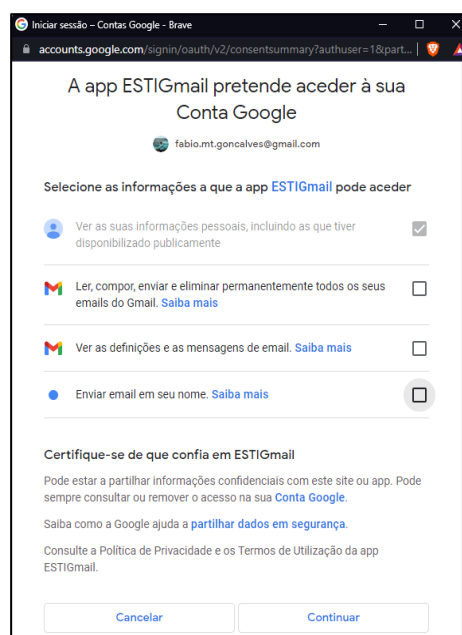
Figura 13 - Componente App.js

### 3.7. Homepage

A página principal do website permite ao utilizador efetuar o login com as suas credenciais da sua conta Gmail e assim conseguir efetuar login com a sua token de autenticação OAuth 2.0, em que ao efetuar login com a sua conta irá ter de dar permissões para que o website possa aceder a certas permissões vindas dos *scopes* definidos na implementação que permitem o website aceder ao email do utilizador e comunicar com sucesso com a *API* do Gmail e fazer os devidos *requests* à mesma.



**Figura 14 - Página Inicial**



**Figura 15 - Pop-up de login com conta gmail (conceder as devidas permissões)**

Ao entrar na página inicial o sistema irá conectar o website ao projeto previamente criado na consola Google através das credenciais também previamente geradas, que neste caso são a ID do Cliente e a API Key, e são também concedidos os scopes que como podemos ver na figura acima o utilizador garante permissão às mesmas para que o website possa aceder à sua conta.

```
const CLIENT_ID = '436969203574-07mg84bon2kd2u4iojum98uqncfgehbe.apps.googleusercontent.com';
const API_KEY = 'AIzaSyCFCwIN9UMULEEazAgMZLvdRpADD57761N';

const DISCOVERY_DOC = 'https://www.googleapis.com/discovery/v1/apis/gmail/v1/rest';

const SCOPES = 'https://mail.google.com/ https://www.googleapis.com/auth/gmail.readonly https://www.googleapis.com/auth/gmail.send https://www.googleapis.com/auth/userinfo.profile';

let tokenClient;
let gapiInited = false;
let gisInited = false;

window.gapi.load('client', initializeGapiClient);

try {
  tokenClient = window.google.accounts.oauth2.initTokenClient({
    client_id: CLIENT_ID,
    scope: SCOPES,
    callback: '', // defined later
  });
  gisInited = true;
} catch (err) {
  window.location.reload();
}

async function initializeGapiClient() {
  await window.gapi.client.init({
    apiKey: API_KEY,
    discoveryDocs: [DISCOVERY_DOC],
  });
  gapiInited = true;
}
```

**Figura 16 - Conexão do website ao projeto no Google console**

Ao clicar no botão de login e após iniciar sessão com as suas credenciais do Gmail e conceder as devidas permissões o sistema irá fazer um request para gerar uma token pertencente ao utilizador para que consiga conectar a sua conta Gmail ao website em correlação ao projeto do Google console após o token ser confirmado, isto é, gerado um novo ou verificado se o utilizador contém já uma e após isso se for bem sucedido o utilizador será redirecionado para a próxima página.

```

function handleAuthClick() {
  tokenClient.callback = async (resp) => {
    if (resp.error !== undefined) {
      throw (resp);
    }
    let response;
    try {
      response = await window.gapi.client.gmail.users.getProfile({
        'userId': 'me',
      });
      console.log("Obj", response);
      console.log("Email", response.result.emailAddress);
      let email = response.result.emailAddress;

      history.push( location: {
        pathname: '/EnviarEmail',
        state: {
          email: email
        }
      });
    } catch (err) {
      console.log(response);
    }
  };

  token();
}


function token(){
  if (window.gapi.client.getToken() === null) {
    tokenClient.requestAccessToken({prompt: 'consent'});
  } else {
    tokenClient.requestAccessToken({prompt: ''});
  }
}

```

**Figura 17 - Autenticação de utilizador através da token**

### 3.8. Enviar Email / Rascunho

[Terminar Sessão](#)



Consultar Emails

Enviar Email / Rascunho

**Emails Enviados**  
Total: 100  
Clique no assunto para visualizar mais informações.

Assunto	Data
Comprovativo de matrícula	Tue, 2 Nov 2021 17:22:35 +0000
Fwd: Relatório - Registo de Tempos de Trabalho	Wed, 1 Sep 2021 14:24:24 +0100
	Fri, 6 Aug 2021 16:38:31 +0100
Candidatura - REPOSITOR (m/f) - Sines	Thu, 5 Aug 2021 10:45:58 +0100
vouchers	Wed, 18 Aug 2021 15:08:05 +0100
imprimir	Wed, 21 Jul 2021 16:30:54 +0100
Ficha de inscrição e documento de Acolhimento Draenrhine	Fri, 6 Aug 2021 13:53:15 +0100

De: fabio.mt.goncalves@gmail.com

Para:

Assunto:

Mensagem:

Escolher Ficheiros Nenhum ficheiro selecionado

Enviar Email

Guardar como Rascunho

**Rascunhos**  
Total: 15  
Clique no assunto para utilizar determinado rascunho.

Assunto	Data
2	Sun, 3 Jul 2022 07:02:37 -0400
	Sun, 3 Jul 2022 07:03:15 -0400
	Mon, 4 Jul 2022 06:36:06 -0400
123333	Sun, 3 Jul 2022 07:01:21 -0400
RaschTest	Sun, 3 Jul 2022 06:53:57 -0400
Fotos	Tue, 11 Sep 2018 20:52:39 +0100
0000	Sun, 3 Jul 2022 07:02:26 -0400
Encomenda atrasada	Tue, 25 May 2021 11:45:32 +0100
Lost Card	Fri, 04 Feb 2022 01:54:54 +0000
Saying Hello	Tue, 15 Mar 2022 16:13:33 +0000

Figura 18 - Interface "Enviar Email / Rascunho"

### 3.9. Tarefa 1 – Enviar Email

O utilizador ao preencher os campos de “Para”, “Assunto” e “Mensagem”, e após clicar no botão de “Enviar Email” o sistema irá utilizar um método “users.messages.send” à API do Gmail em que primeiramente terá de ser criado um body (corpo da mensagem), em que consistem em juntar os campos para enviar o método e a API do Gmail receber em formato de JSON em que para isso será passado os valores das input boxes para “From”, “To”, “Subject” e a mensagem que terá de ser codificada devido às normas OAuth2.0 para que possa assim ser enviada através da API do Gmail em que após isso o utilizador irá ser notificado por uma popup gerada pelo browser a dizer que o email foi bem enviado com sucesso.

```

function send(e) {

    e.preventDefault();
    const form = e.target;
    const emailTo = form.elements["emailTo"].value;
    const subject = form.elements["subject"].value;
    const messageContext = form.elements["message"].value;

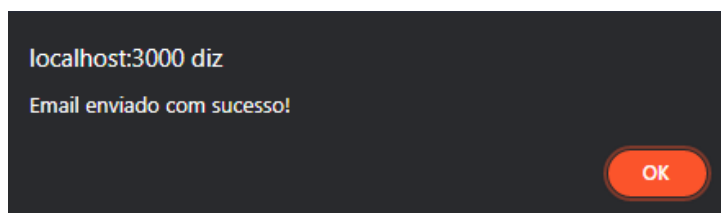
    const message =
        "From: " + email + "\r\n" +
        "To: " + emailTo + "\r\n" +
        "Subject: " + subject + "\r\n\r\n" + messageContext;

    const encodedMessage = btoa(message).replace(/ /g, '%20').replace(/\\//g, '%5C').replace(/_./g, '%5F');

    window.gapi.client.gmail.users.messages.send({
        userId: 'me',
        resource: {
            raw: encodedMessage
        }
    }).then(function () {
        console.log('Mensagem enviada.');
        //document.getElementById("table-sent").remove();
        //showSentEmails();
        window.alert("Email enviado com sucesso!");
        clear();
    }, function (error) {
        console.log(error);
        window.alert("Erro ao enviar email.");
    });
}

```

**Figura 19 - Função para enviar email**



**Figura 20 - Popup (Email enviado com sucesso)**

### 3.10. Tarefa 2 – Criar Rascunho

A segunda tarefa é relativamente semelhante à primeira tarefa visto que requer os mesmos parâmetros e a mensagem será codificada da mesma forma a única diferença que requer o uso deste request à API do Gmail é o uso do método “users.drafts.create”, após o bem sucedido, tal e qual como anteriormente, o browser irá disponibilizar uma mensagem popup para o utilizador receber a indicação de que o rascunho foi guardado com sucesso.



```
function draft() {

    let emailTo = document.getElementById( elementId: "emailTo").value;
    let subject = document.getElementById( elementId: "subject").value;
    let messageContext = document.getElementById( elementId: "message").value;

    const message =
        "From: " + email + "\r\n" +
        "To: " + emailTo + "\r\n" +
        "Subject: " + subject + "\r\n\r\n" + messageContext;

    const encodedMessage = btoa(message).replace( searchValue: /\+/g, replaceValue: '-' ).replace( searchValue: /\//g, replaceValue: '_' ).replace( searchValue: /%$/ , replaceValue: '' );

    window.gapi.client.gmail.users.drafts.create( params: {
        userId: 'me',
        message: {
            raw: encodedMessage
        }
    }).then(function () {
        console.log("Rascunho Guardado.");
        window.alert("Rascunho guardado com sucesso!");
        clear();
    }, function (error) {
        console.log(error);
        window.alert("Erro ao guardar rascunho.");
    });
}
```

Figura 21 - Função para criação de um rascunho

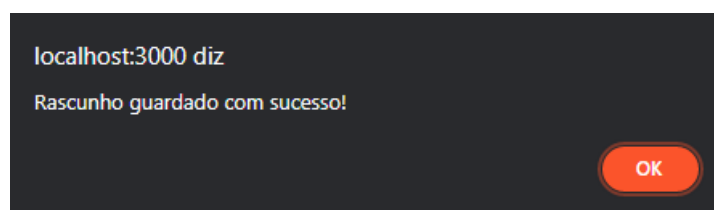


Figura 22 - Popup (Rascunho guardado com sucesso)

### 3.11. Tarefa Extra – Visualizar / Utilizar Rascunhos

O utilizador ao fazer login e entrar na devida página após o mesmo, poderá visualizar todos os rascunhos que tem criados e ao clicar em cima do assunto faz com que os campos de “Para”, “Assunto” e “Mensagem” fiquem automaticamente preenchidos com os valores correspondentes ao devido rascunho clicado, assim permite ao utilizador facilmente enviar ou editar rascunhos previamente criados.

O método de visualizar rascunhos primeiramente é necessário obter uma lista de ids utilizando o método “users.messages.list” em que cada id irá corresponder a uma label do tipo “DRAFT” que após ser gerado irá correr um .each em JQuery com o método “users.messages.get” e assim associar um id a o respetivo rascunho que permite assim executar a próxima função “appendMessageRowDraft”, em que consiste em incrementar um count cada vez que entra nessa função, ou seja, irá assim calcular o número total de rascunhos e após isso irá então criar uma tabela com o assunto e a data disponíveis logo para o utilizador visualizar, mas só após o clique no assunto irá então preencher o formulário com os devidos campos automaticamente.

Após a criação da tabela utilizando a biblioteca JQuery, foi ainda utilizado o mesmo para obter um clique em determinada linha da tabela, estas pertencendo à coluna “Assunto”. Esse clique é iniciado através do id que foi gerada previamente com a tabela em que cada linha da tabela terá um id do estilo “draft- ‘message-id’”, em que o ‘message-id’ será o id do rascunho, portanto cada id será unicamente diferente e cada um pertencendo a um rascunho diferente

Após o clique de uma das linhas da tabela pertencente à coluna “Assunto” irão então ser executadas as funções `getHeader`, que irá indexar a string “To” que irá corresponder ao “Para”, isto é para quem o rascunho foi originalmente criada para ser enviado, e o mesmo para o “Subject” que irá preencher o campo “Assunto” do formulário e por fim a função “`getBody`” que ao contrário das tarefas um e dois irá então decodificar a mensagem para que esta possa ser recebida no campo de “Mensagem” no formulário para que o utilizador a possa visualizar e editar se este mesmo pretender.

Ainda a referir que a função de criação da tabela que representa os rascunhos irá ser executada logo após a página ser carregada, isto é, a interface “Enviar Email / Rascunho” através do código que podemos visualizar na figura abaixo.

```
window.onload = showDrafts();
```

Figura 23 - Executar função "showDrafts()" após interface ser inicializada

```
function showDrafts() {
  const request = window.gapi.client.gmail.users.messages.list({
    'userId': 'me',
    'labelIds': 'DRAFT',
  });

  request.execute(function(response : Error | null ) {
    $.each(response.messages, function() {
      const messageRequest = window.gapi.client.gmail.users.messages.get({
        'userId': 'me',
        'id': this.id
      });

      messageRequest.execute(appendMessageRowDraft);
    });
  });
}
```

Figura 24 - Função para mostrar rascunhos criados

```

function appendMessageRowDraft(message) {

    console.log(message.payload)

    countDraft++;
    document.getElementById( elementId: "counterDraft").innerHTML = "Total: " + countDraft;

    $('#table-draft').append(
        '<tr>\n
        <td id="draft-' + message.id + '" >' + getHeader(message.payload.headers, index: 'Subject') + '</td>\n
        <td>' + getHeader(message.payload.headers, index: 'Date') + '</td>\n
        </tr>'
    );

    $('#draft-' + message.id).click(function(){
        document.getElementById( elementId: "emailTo").value = getHeader(message.payload.headers, index: 'To');
        document.getElementById( elementId: "subject").value = getHeader(message.payload.headers, index: 'Subject');
        document.getElementById( elementId: "message").value = getBody(message.payload);
    });
}

```

Figura 25 - Criação da tabela rascunhos e click (jQuery)

```

function getHeader(headers, index) {
    let header = '';

    $.each(headers, function(){
        if(this.name === index){
            header = this.value;
        }
    });
    return header;
}

function getBody(message) {
    let encodedBody;
    if(typeof message.parts === 'undefined')
    {
        encodedBody = message.body.data;
    }
    else
    {
        encodedBody = getHTMLPart(message.parts);
    }
    encodedBody = encodedBody.replace(/-/g, '+').replace(/_/g, '/').replace(/\\s/g, '');
    return decodeURIComponent(escape(window.atob(encodedBody)));
}

```

Figura 26 - Funções getHeader() e getBody()

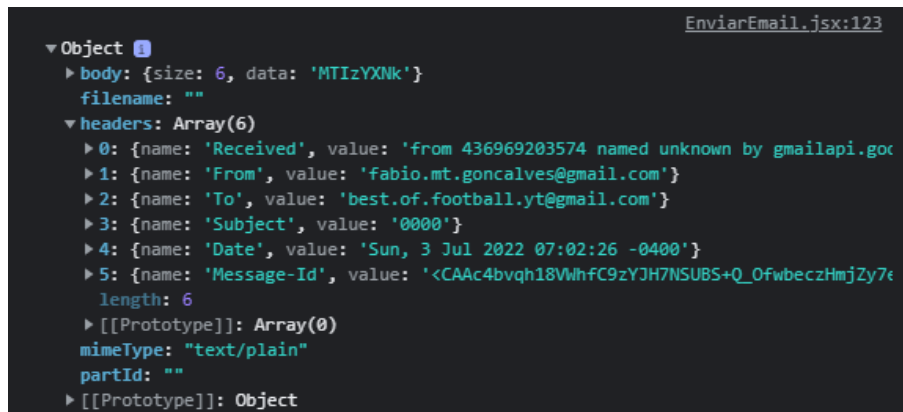


Figura 27 - message.payload

### 3.12. Tarefa Extra – Consultar Emails Enviados

Esta tarefa consiste também em todas as referências mostradas na tarefa extra descrita acima (Visualizar / Utilizar Rascunhos) uma das únicas exceções foi que ao utilizar o método da API do Gmail “users.messages.list” em que cada id irá corresponder também a uma só que neste caso será a uma label id “SENT” que irá devolver os ids dos emails enviados e assim adiante que após isso também será utilizado o método “users.messages.get” para após isso realizar o mesmo procedimento de criar a tabela com esses ids.

A grande diferença desta tarefa é que após clicar no determinado assunto irá ser mostrada uma popup em que o utilizador poderá consultar mais informações em que poderá visualizar para qual o destinatário da mensagem era e o conteúdo da mensagem.

```

window.onload = showSentEmails();
  
```

Figura 28 - Executar função "showSentEmails()" após interface ser inicializada

```

function showSentEmails() {
  const request = window.gapi.client.gmail.users.messages.list({
    'userId': 'me',
    'labelIds': 'SENT',
  });

  request.execute(function(response :Error| null ) {
    $.each(response.messages, function() {
      const messageRequest = window.gapi.client.gmail.users.messages.get({
        'userId': 'me',
        'id': this.id
      });

      messageRequest.execute(appendMessageRowSent);
    });
  });
}
  
```

Figura 29 - Função para mostrar emails enviados

```

function appendMessageRowSent(message) {

    console.log(message.payload)

    //array[countSent] = getBody(message.payload);
    //console.log(array)

    countSent++;

    document.getElementById( "counterSent").innerHTML = "Total: " + countSent;

    $('#table-sent').append(
        '<tr>\n
        <td id="sent-message-' + message.id + '" >' + getHeader(message.payload.headers, index: 'Subject')+</td>\n
        <td>' + getHeader(message.payload.headers, index: 'Date')+</td>\n
        </tr>'
    );

    $('#sent-message-' + message.id).click(function(){
        $('.popup').show();
        document.getElementById( "contentEmailTo").innerHTML = getHeader(message.payload.headers, index: 'To');
        document.getElementById( "content").innerHTML = getBody(message.payload);
    });

    $('.popup').click(function(){
        $('.popup').hide();
    });
    $('.popupCloseButton').click(function(){
        $('.popup').hide();
    });
}

```

Figura 30 - Criação da tabela de emails enviados e popup (jQuery)



Figura 31 - Popup de consultar email enviado

### 3.13. Expirar da sessão

O utilizador ao tentar fazer “refresh” ao browser ou até mesmo fechá-lo enquanto se encontra na página do website o browser irá mostrar uma popup em que diz ao utilizador que poderá perder alterações e pergunta-lhe se ele quer mesmo continuar a realizar aquela ação ou deseja cancelar, isto é porque ao atualizar a página ou fechá-la o utilizador irá perder acesso às credenciais que estão ligada à token daquela sessão e então terá de realizar o início de sessão na sua conta Gmail novamente.

O sistema implementado desta maneira demonstra também segurança no que toca à privacidade e recolha de dados do utilizador visto que se o mesmo se esqueça da conta com sessão iniciado no website, a sessão irá automaticamente expirar visto que o website não guarda esse tipo de informações devido ao facto de preservar a privacidade do utilizador.

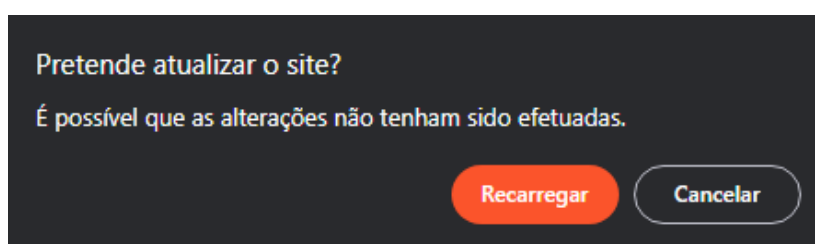


Figura 32 - Popup de confirmação após tentar atualizar ou fechar a janela do browser

Caso o utilizador clique em “Recarregar” a página ou até mesmo se estiver na página inicial e tentar ir para a página “EnviarEmail” sem ter sessão iniciada, por exemplo, o utilizador ao encontrar-se na página “localhost:3000” e tentar aceder à página “localhost:3000/enviaremail” sem ter sessão iniciada irá se deparar com a interface mostrada na figura abaixo.



Figura 33 - Interface de nenhuma sessão iniciada

## 4. Avaliação das Interfaces / Testes com Utilizadores

Neste tópico são explicadas as metodologias e os procedimentos executados nos testes com utilizadores que permitiram avaliar o grau de usabilidade do sistema Web.

### 4.1. Guião

O guião que foi apresentado aos utilizadores foi com a seguinte estrutura:

- Apresentação do utilizador que vai realizar o teste;
- Pedir ao utilizador para efetuar login com a sua conta Gmail;
- Pedir ao utilizador para enviar um email;
- O utilizador verifica se o email foi enviado através do site oficial do Gmail;
- Pedir ao utilizador para criar um rascunho;
- O utilizador verifica se o rascunho foi criado através do site oficial do Gmail;
- Indicar ao utilizador para consultar um email que tenha sido enviado por o mesmo;
- Indicar ao utilizador para utilizar um rascunho que tenha sido previamente criado e enviar o mesmo como email.

#### I. Número de Utilizadores: 2;

**Local:** Residência do utilizador;

**Gravação dos dados:** Utilizador 1: Através do Microsoft Teams, em que foi admitido controlo do meu computador ao utilizador.

Utilizador 2: Presencialmente na casa do utilizador, em que o utilizador utilizou o meu computador para realizar o teste.

(Ambas as gravações foram efetuadas por captura de ecrã e nenhum dos utilizadores se sentiu confortável em gravar por videochamada).

#### II. Ambos os utilizadores acharam a interface bastante simples e fácil de enganar e que quem está habituado a utilizar um serviço email regularmente não vai ter dificuldades nenhuma em se adaptar.

No fim, quando foram questionados acerca de melhorias no sistema ambos disseram que existem “bugs” no que toca ao facto de ao clicar num rascunho por vezes a mensagem que aparece no formulário vem desformatada, e daí terem o trabalho de apagar essas palavras a mais, o mesmo acontece por vezes o campo de email vindo do rascunho, essas falhas podem originar devido ao facto da decodificação estar feita para tipos de emails com um corpo específico.

Utilizador	Tarefa 1		Tarefa 2		Tarefa 3		Tarefa 4	
	Tempo (s)	Clicks	Tempo (s)	Clicks	Tempo (s)	Clicks	Tempo (s)	Clicks
1	20.06	4	17.02	5	13.79	2	10.46	2
2	16.73	5	12.57	4	10.84	1	32.85	8

Tabela 3 - Resultados dos testes com utilizadores

**Tarefa 1** – Enviar Email

**Tarefa 3** – Consultar emails enviados

[Link do teste com utilizador 1](#)

**Tarefa 2** – Criar Rascunho

**Tarefa 4** – Utilizar um rascunho

[Link do teste com utilizador 2](#)

## **5. Conclusões Finais**

Por fim, posso concluir que foi um trabalho bem executado, houve bastantes etapas a superar, talvez a escolha da API do Gmail tenha dificultado um pouco a execução desta segunda parte do projeto visto que foi bastante tempo a tentar perceber como sincronizar corretamente a token de utilizador por autenticação OAuth entre a API do Gmail e o meu projeto, houve bastantes etapas superados em que me forneceu mais conhecimentos que não tinha ao trabalhar com React e mesmo também com JavaScript, CSS e Html.

Portanto tenho a concluir que ganhei bastantes conhecimentos novos não só ao realizar este trabalho como também ao decorrer da unidade curricular e que forem aqui que a maioria desses conhecimentos foram aplicados, e por fim posso concluir que foi um trabalho bem executado no fim em que consegui concluir o objetivo final.