Relatório Técnico da Aplicação SupaSocial

Ângelo Neto [1230439], Luís Oliveira [1231002]

Instituto Superior de Engenharia do Porto {1230439,Anglnet,
1231002,1231002Luis}@github.com
https://github.com/1231002Luis/DSSMV_ProjectReact_1231002_1230439

Abstract. This project involves the development of a mobile application simulating a social network for Android and iOS devices. Users can create accounts and personalize their profiles by adding personal information, an avatar, and a biography. Interaction between users is facilitated through posts, which can be commented on, shared, and liked. The application also includes real-time notifications to keep users informed about activities related to their posts. This platform aims to deliver an engaging and dynamic social networking experience.

1 Problema

Este trabalho consiste em desenvolver uma aplicação que simula o contexto de uma rede social, para dispositivos móveis Android e IOS. Os utilizadores podem criar uma conta e editar o seu perfil com os seus dados pessoais, um avatar e uma biografia. A interação entre utilizadores é feita através de publicações, que podem ser comentadas, partilhadas e receber um "like". É possível obter notificações sobre as publicações realizadas pelo utilizador em tempo real.

1.1 User Stories

Como utilizador quero:

- Editar o meu perfil;
- Criar, gostar, comentar, partilhar, editar ou apagar um post;
- Receber notificações sobre os meus posts.

2 Análise

2.1 Modelo de domínio

No modelo de domínio estão representados os conceitos fundamentais da aplicação. O utilizador é a entidade central, visto que apresenta relações com os restantes conceitos. Cada utilizador pode efetuar vários comentários, receber várias notificações, postar ou gostar de várias publicações. A publicação pode receber vários gostos.

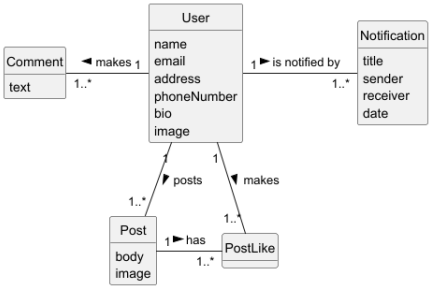

Fig. 1: Modelo de domínio

2.2 Requisitos não funcionais

- Usabilidade
  - Interface gráfica.
- Fiabilidade
  - Não especificado.
- Desempenho
  - Não especificado.
- Sustentabilidade
  - Plataforma Android
- Restrições de Design
  - Não especificado.
- Restrições de Implementação
  - Linguagem React Native
- Restrições de Interface
  - Interface gráfica Android e IOS

2.3 Requisitos funcionais

- Funcionalidades

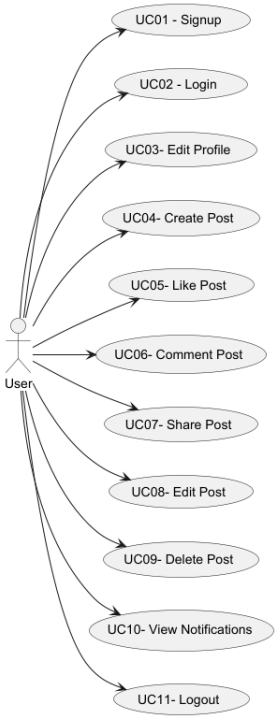

Fig. 2: Diagrama de casos de uso

Foram estabelecidos os seguintes casos de uso:

– Utilizador
  - UC01- Sign up.
  - UC02 - Login.
  - UC03 - Edit Profile.
  - UC04 - Create Post.
  - UC05 - Like Post.
  - UC06 - Comment Post.
  - UC07- Share Post.
  - UC08 - Edit Post.
  - UC09 - Delete Post.
  - UC10 - View Notifications
  - UC11 - Logout

2.4 Especificação UC01

Descrição	Fazer login na conta
Pré-condição	Estar na página de login
Pós-condição	Login efetuado
Caminho principal	1. O utilizador seleciona a opção de login 2. O sistema pede as credenciais 3. O utilizador introduz as credenciais 4. O sistema valida a conta e averigua se existe 5. Retorna sucesso.
Caminho alternativo	1. As credenciais estão erradas ou conta não existe. 2. Retorna erro.

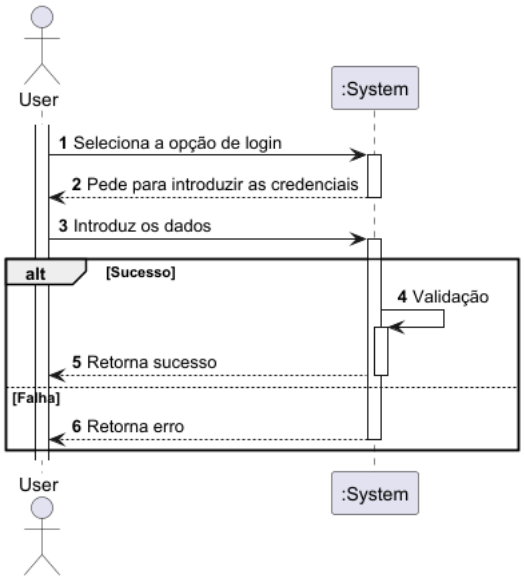

Fig. 3: SSD-UC01 : Login

2.5 Especificação UC04

Descrição	Criar publicação

Pré-condição	Estar no home page do utilizador
Pós-condição	Publicação efetuada
Caminho principal	1. O utilizador seleciona a opção de criar publicação 2. O sistema pede os dados a publicar 3. O utilizador introduz o texto 4. **Opcional**: O utilizador introduz imagem ou vídeo 5. Retorna sucesso.
Caminho alternativo	1. O utilizador não introduziu texto, imagem ou vídeo. 2. Retorna erro.

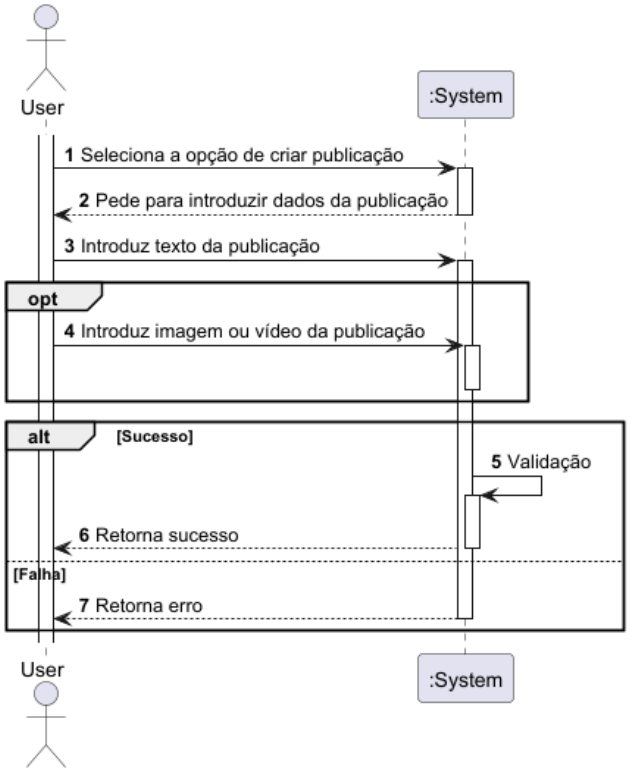

Fig. 4: SSD-UC04 : Create Post

2.6 Especificação UC09

Descrição	Eliminar publicação
Pré-condição	Selecionar publicação a pagar
Pós-condição	Publicação eliminada
Caminho principal	1. O utilizador seleciona a publicação a eliminar 2. O sistema retorna a publicação 3. O utilizador seleciona opção para apagar publicação 4. O sistema pede confirmação 5. O utilizador confirma 6. Retorna sucesso.
Caminho alternativo	1. O utilizador não introduziu texto, imagem ou vídeo. 2. Retorna erro.

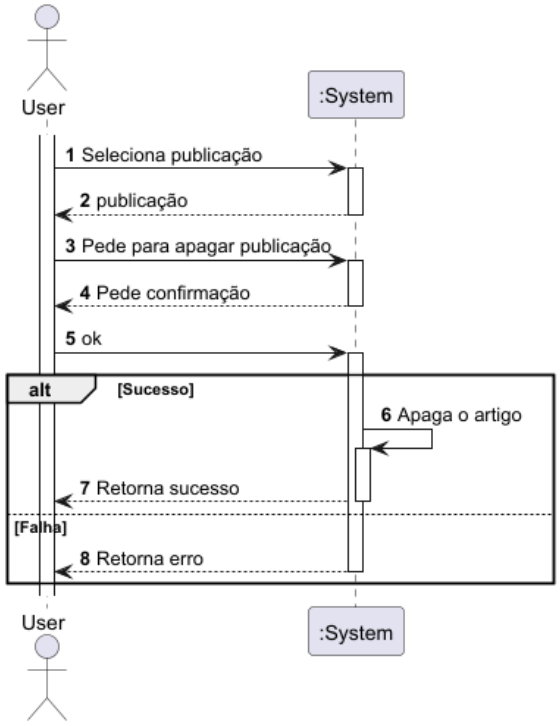

Fig. 5: SSD-UC09 : Delete Post

3 Design

3.1 Arquitetura

Fig. 6: Arquitetura física

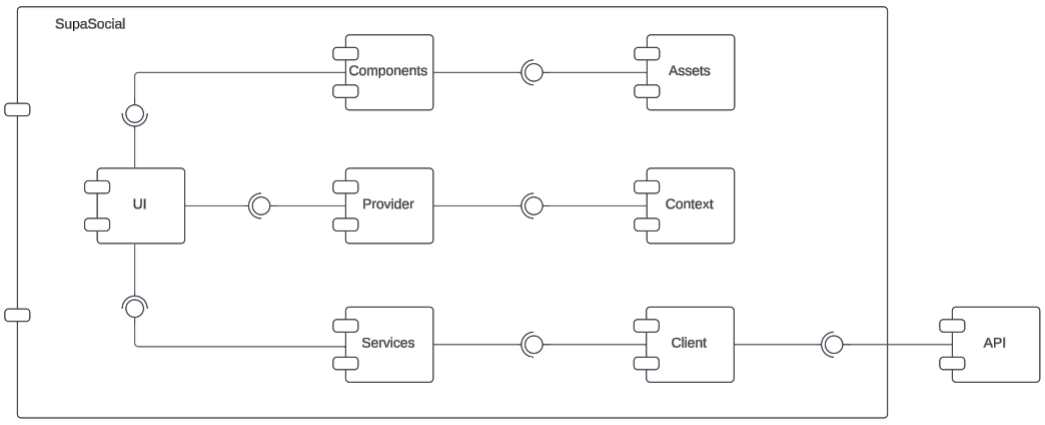

Fig. 7: Arquitetura lógica

3.2 Organização do código

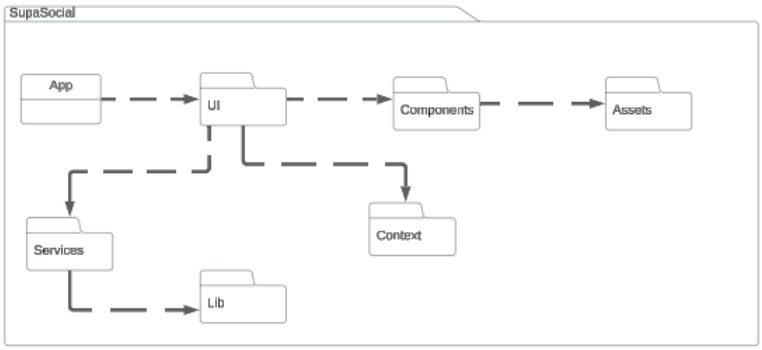

Fig. 8: Organização do código

3.3 Diagramas de sequência

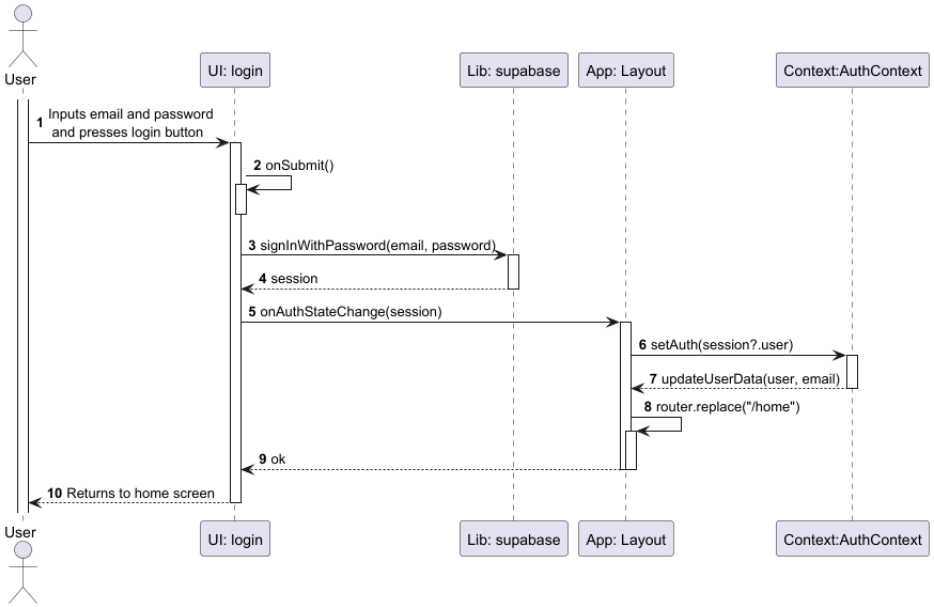

Fig. 9: US01: Login

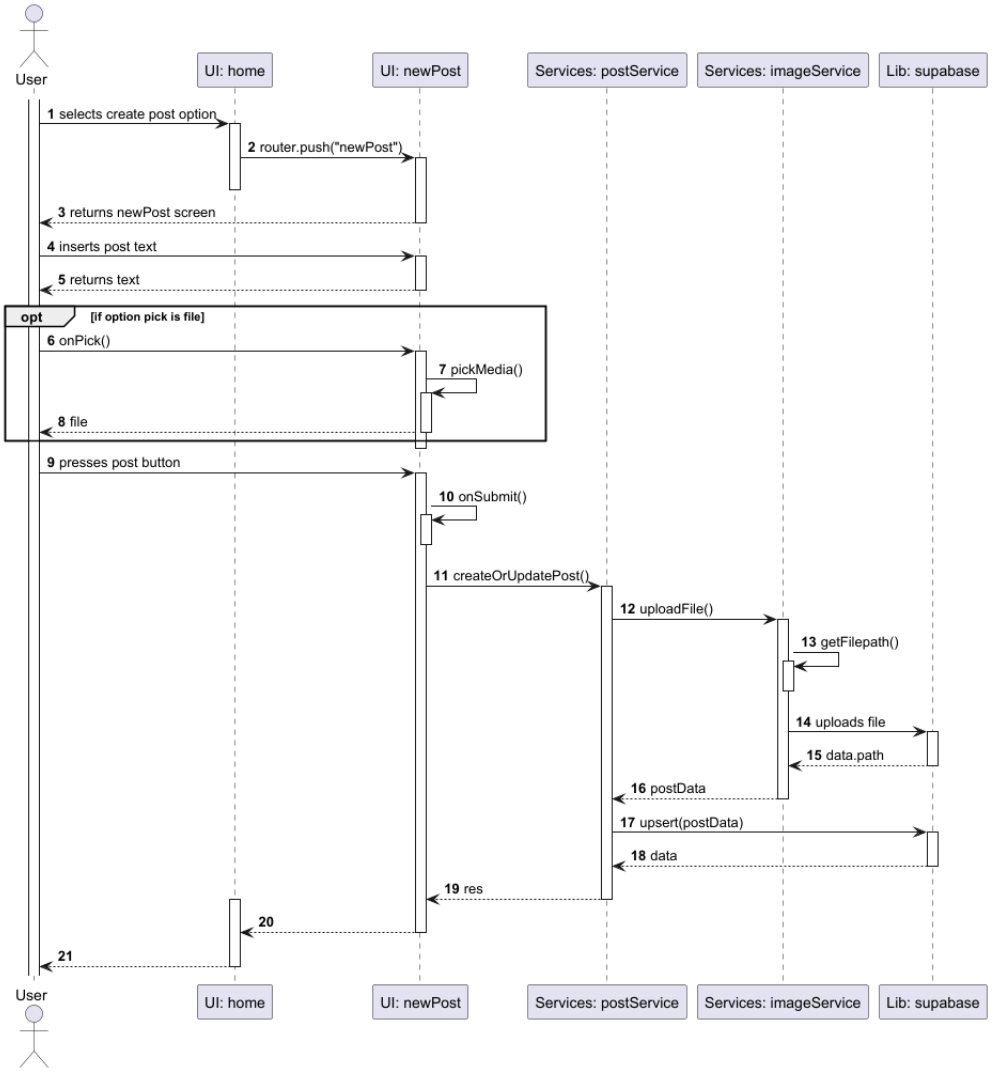

Fig. 10: US04: Criar Publicação

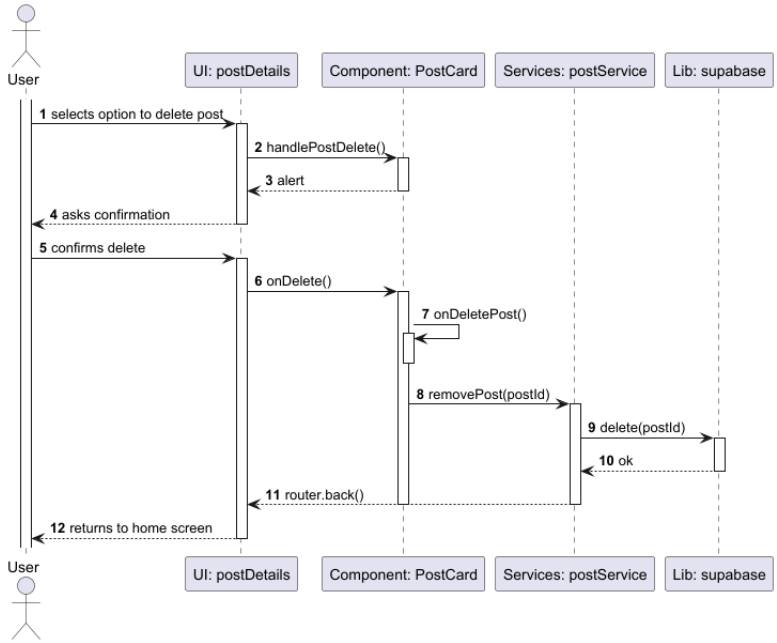

Fig. 11: US09: Apagar Publicação

3.4 User Interface

Fig. 12: Welcome Screen

Fig. 13: Sign up Screen

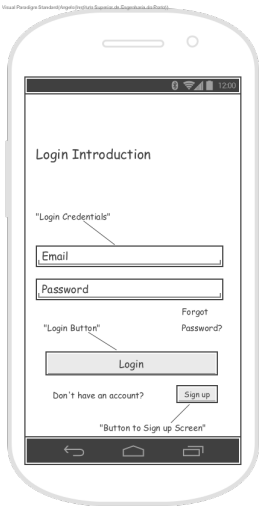

Fig. 14: Login Screen

O menu inicial da aplicação consiste numa interface com uma imagem representativa e o nome da rede social, incluindo dois botões, um para criar conta se o utilizador ainda não se registou, e outro botão para fazer login. Ao selecionar o botão de registo, o utilizador é redirecionado para um menu onde pode escrever

as credenciais da sua conta , bem como voltar para o menu de Login no caso de já ter uma conta criada. Ao selecionar o botão de login, o utilizador pode escrever as suas credenciais para poder entrar na sua conta.

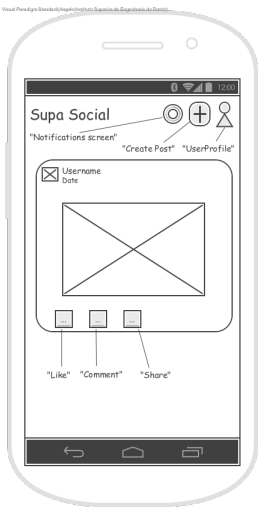

Fig. 15: Home Screen

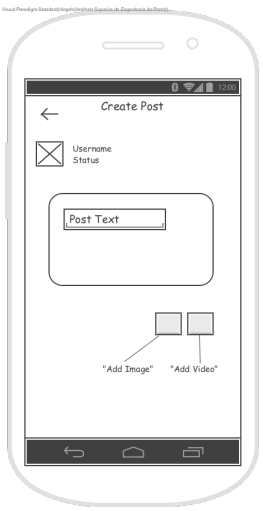

Fig. 16: Create Post Screen

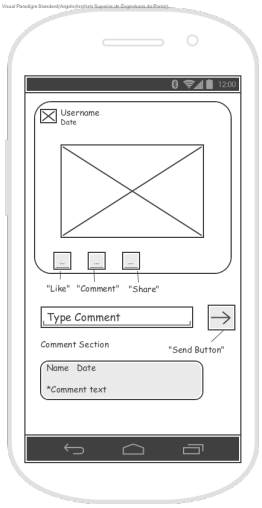

Fig. 17: Comment Screen

Após a autenticação do utilizador, a aplicação é redirecionada para o menu principal, que apresenta 3 botões no header: Abrir notificações, criar nova publicação e abrir perfil de utilizador. Na parte principal do menu, surgem as pub-

licações que foram efetuadas pelo utilizador ou outros utilizadores, em forma de pilha, ou seja as publicações mais recentes são colocadas em cima. Ao selecionar o botão de criar publicação, o utilizador é remetido a página de publicação, que contêm um body para o texto a enviar e dois botões para inserir imagem ou vídeo. Ao selecionar uma publicação no menu principal, é aberta a página dos comentários do post, onde é possível enviar comentários.

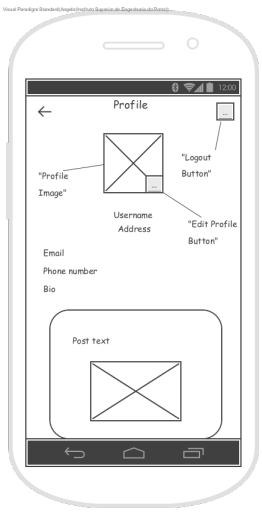

Fig. 18: Profile Screen

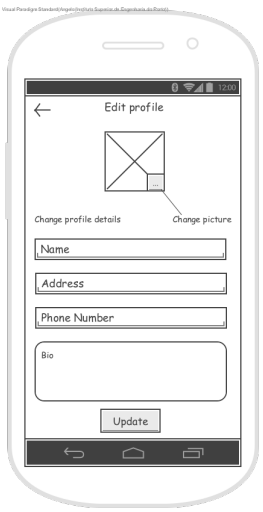

Fig. 19: Edit Profile Screen

Se a opção de perfil for selecionada, o menu de perfil é iniciado. A partir deste menu, é possível visualizar e alterar a foto de perfil. Também é possível ver as informações do utilizador e os posts do mesmo. A opção de logout da conta, está presente no header do menu. Ao selecionar o botão de editar perfil, o utilizar é redirecionado para o menu de edição de perfil, onde pode atualizar a foto e os seus atributos.

4 Implementação

– Login

```
1    const onSubmit = async()=>{
2        if(!emailRef.current || !passwordRef.current){
3            Alert.alert('Login', "Please fill all the fields");
4            return;
5        }
6
7        let email = emailRef.current.trim();
```

```
8        let password = passwordRef.current.trim();
9
10        setLoading(true);
11       const {error} = await supabase.auth.signInWithPassword({
12            email,
13            password,
14        });
15
16        setLoading(false);
17
18        console.log('error', error);
19        if(error){
20            Alert.alert('Login', error.message);
21        }
22    }
```

Este código é executado quando o utilizador pressiona no botão Login. Para o login funcionar, é lido o texto introduzido nos campos email e password e guardado em duas variáveis. Essas variáveis vão posteriormente ser usadas como parametros da função signInWithPassword da supabase. Esta função é uma função assíncrona que retorna uma Promise. No caso da Promise ser reijeitada, é retornado um erro e esse erro é apresentado ao utilizador.

```
1        supabase.auth.onAuthStateChange((_event, session) => {
2   console.log("session user: ", session?.user);
3
4   if (session) {
5    setAuth(session?.user);
6    updateUserData(session?.user, session?.user?.email);
7    router.replace("/home");
8   } else {
9    setAuth(null);
10    router.replace("/welcome");
11   }
12  });
```

Complementando o código anterior, a função onAuthStateChange é executada sempre que é alguma mudança no estado de autenticação da aplicação. Esta função está à "escuta" dessa mudança. Quando executada define globalmente que um utilizador está autenticado com o hook setAuth e define os dados do utilizador com a função updateUserData. O utilizador é posteriormente redirecionado para o menu principal. Para esse efeito é utilizado o hook useRouter que permita uma navegação baseada em ficheiros.

– Criar publicação

```
1    const pickMedia = async (useCamera, isImage) => {
2  let mediaConfig = {
3   mediaTypes: ImagePicker.MediaTypeOptions.Images,
4   allowsEditing: true,
5   aspect: [4, 3],
6   quality: 0.7,
7  };
8
9  if (!isImage) {
10   mediaConfig = {
11    mediaTypes: ImagePicker.MediaTypeOptions.Videos,
12    allowEditing: true,
13   };
14  }
15
16  let result;
17
18  if (useCamera) {
19   result = await ImagePicker.launchCameraAsync(mediaConfig);
20  } else {
21   result = await ImagePicker.launchImageLibraryAsync(mediaConfig);
22  }
23
24  if (!result.canceled) {
25   setFile(result.assets[0]);
26  }
27 };
```

O código apresentado acima é executado quando o utilizador escolhe inserir um ficheiro na publicação. Para esse efeito é utilizada a library ImagePicker que permite utilizar a camera diretamente na aplicação (a partir da função launchCameraAsync()) ou escolher um ficheiro da galeria (a partir da função launchImageLibraryAsync()).

```
1    export const createOrUpdatePost = async (post) => {
2 try {
3  if (post.file && typeof post.file == "object") {
4   let isImage = post?.file?.type == "image";
5   let folderName = isImage ? "postImages" : "postVideos";
6   let fileResult = await uploadFile(folderName, post?.file?.uri, isImage);
7   if (fileResult.success) post.file = fileResult.data;
8   else {
9    return fileResult;
```

```
  }
 }

 const { data, error } = await supabase
  .from("posts")
  .upsert(post)
  .select()
  .single();

 if (error) {
  console.log("createPost error: ", error);
  return { success: false, msg: "Could not create your post" };
 }

 return { success: true, data: data };
} catch (error) {
 console.log("createPost error: ", error);
 return { success: false, msg: "Could not create your post" };
}
};
```

Após o utilizador pressionar o botão Create Post, é feito um pedido a api supabase de upsert (cria um utilizador novo se nao existir nenhum com aquele id, e atualiza se ja existir). Para esse efeito, se o post possuir um ficheiro é executado o seguinte:

```
  export const uploadFile = async (folderName, fileUri, isImage = true) => {
try {
 let fileName = getFilePath(folderName, isImage);
 const fileBase64 = await FileSystem.readAsStringAsync(fileUri, {
  encoding: FileSystem.EncodingType.Base64,
 });
 let imageData = decode(fileBase64);
 let { data, error } = await supabase.storage
  .from("uploads")
  .upload(fileName, imageData, {
   cacheControl: "3600",
   upsert: false,
   contentType: isImage ? "image/*" : "video/*",
  });

 if (error) {
  console.log("file upload error: ", error);
```

```
18   return { success: false, msg: "Could not upload media" };
19  }
20
21  return { success: true, data: data.path };
22 } catch (error) {
23  console.log("file upload error: ", error);
24  return { success: false, msg: "Could not upload media" };
25 }
26 };
27
28   export const getFilePath = (folderName, isImage) => {
29   return `/${folderName}/${new Date().getTime()}${isImage ? ".png" : ".mp4"}`;
30   };
```

Para carregar o ficheiro na storage da supabase é necessário o caminho do ficheiro que é criado a partir da função getFilePath que utiliza a data de criação para produzir nome de ficheiros únicos. Na documentação oficial da supabase é dito explicitamente que para React Native é necessário enviar o ficheiro em formato binário. Para esse efeito o ficheiro foi transformado em ficheiro Base64 (sequência de caracteres ASCII) para depois ser convertido para formato binário.

5 Testes

UC02 - Login

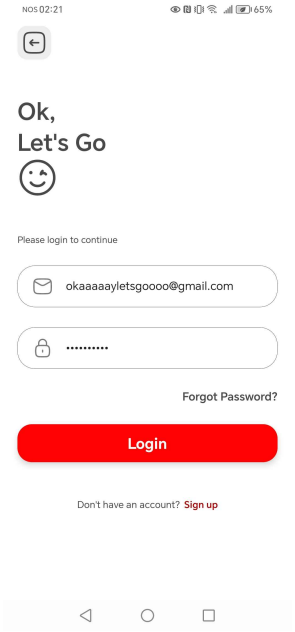

Fig. 20: Teste de login

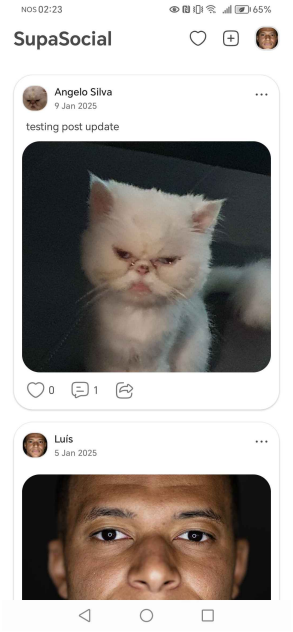

Fig. 21: Teste de login sucesso

Ao pressionar no botão login o utilizador é redirecionado para o menu principal se as credênciais tiverem certas, com acesso à sua conta.

UC04 - Criar publicação

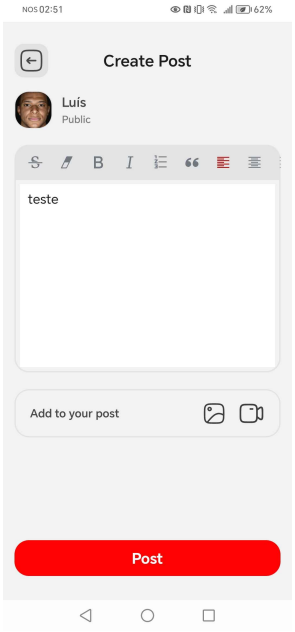

Fig. 22: Teste de adicionar publicação

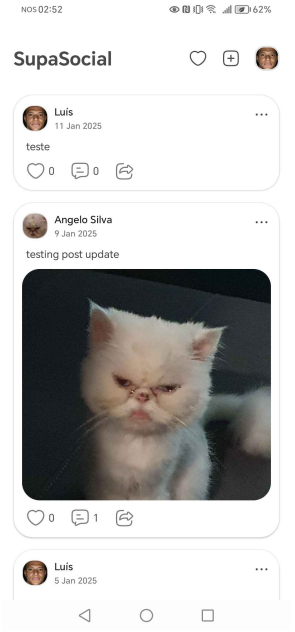

Fig. 23: Teste de adicionar publicação sucesso

Quando se pressiona no botão Post, o utilizador é redirecionado para o menu principal, onde pode encontrar a sua nova publicação.

UC09 - Apagar publicação

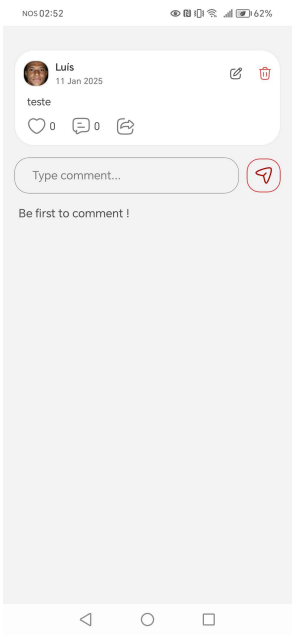

Fig. 24: Teste de apagar publicação

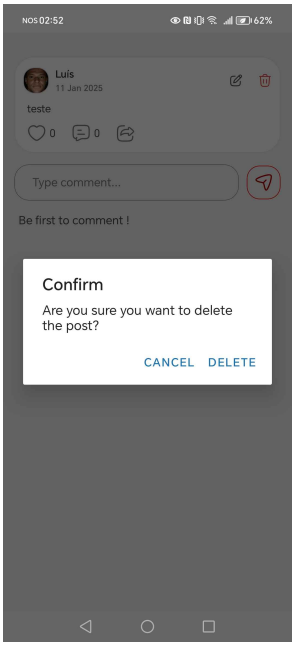

Fig. 25: Teste de apagar publicação (confirmação)

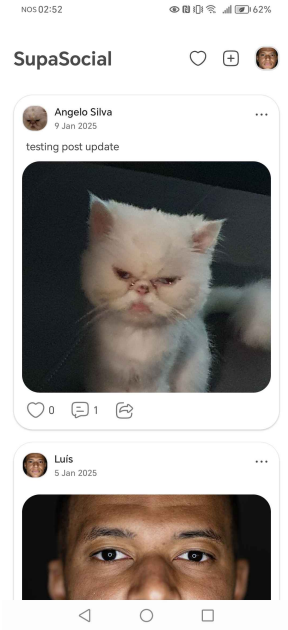

Fig. 26: Teste de apagar publicação sucesso

Se a publicação pertencer ao utilizador, o mesmo tem opção de a apagar, tendo um processo muito semelhante ao do publicar publicação. O utilizador é redirecionado para o menu principal e a publicação é apagada.

6 Conclusão

Ao longo do desenvolvimento deste projeto, adquiriu-se um conhecimento significativo sobre a linguagem JavaScript, o framework React Native e o desenvolvimento de aplicações móveis. Foi atribuída uma forte prioridade à análise e ao design do software antes da implementação, o que facilitou de forma clara o desenvolvimento do sistema. No entanto, surgiram dificuldades consideráveis na implementação da funcionalidade de inclusão de ficheiros nas publicações, bem como na atualização dos dados das mesmas. É igualmente relevante referir que a funcionalidade de partilhar publicações não está a funcionar de forma integral; embora o utilizador consiga partilhar uma publicação, os ficheiros associados a esta não são devidamente incluídos na partilha.