Layout

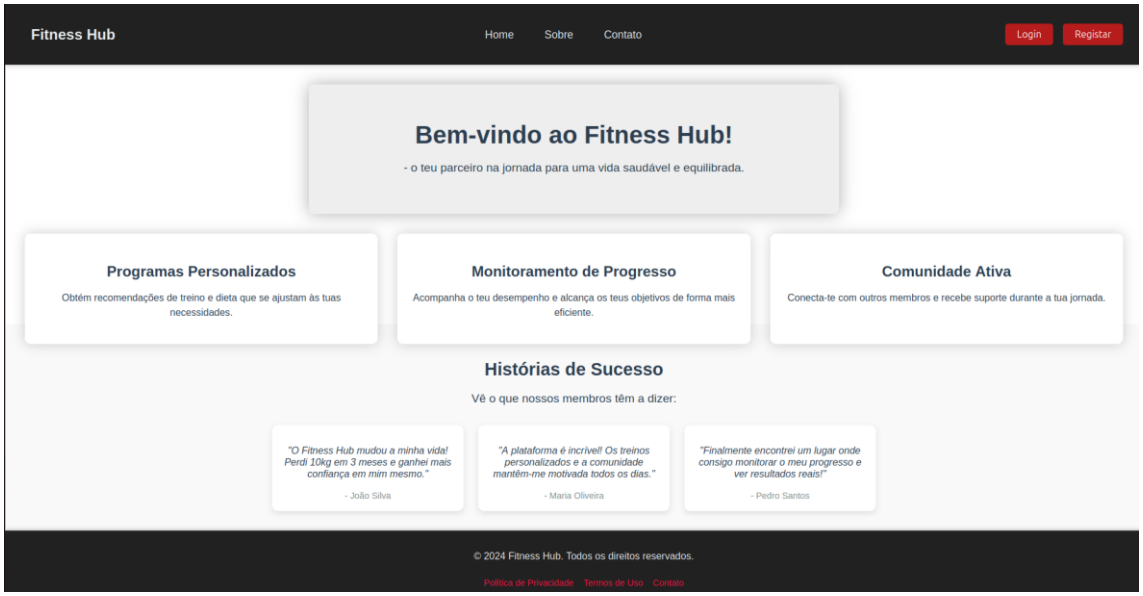

Fig1- HomePage

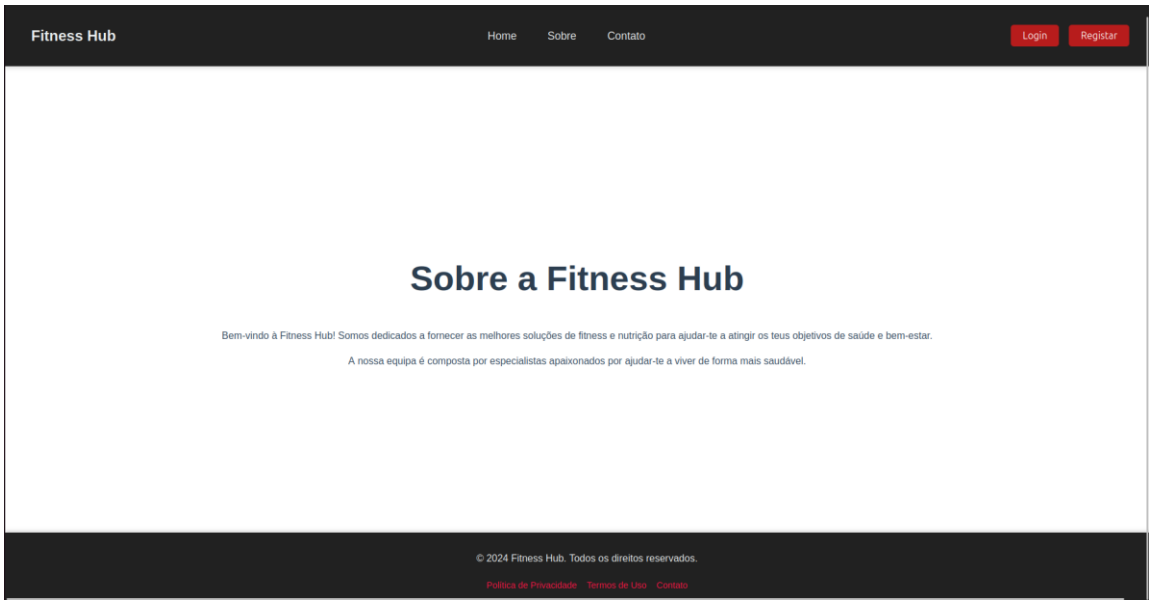

Fig2- Página Sobre o site/empresa

Trabalho feito por:
Alexandre Santos 71522,
Pedro Antunes 75409

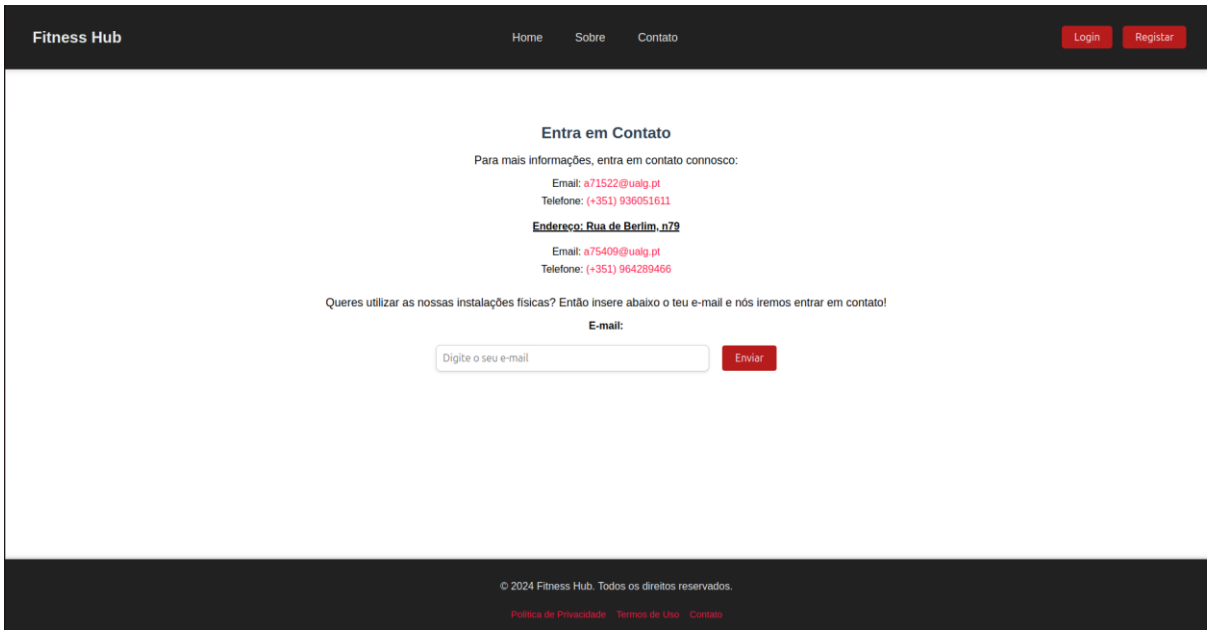

Fig3- Página contactos.

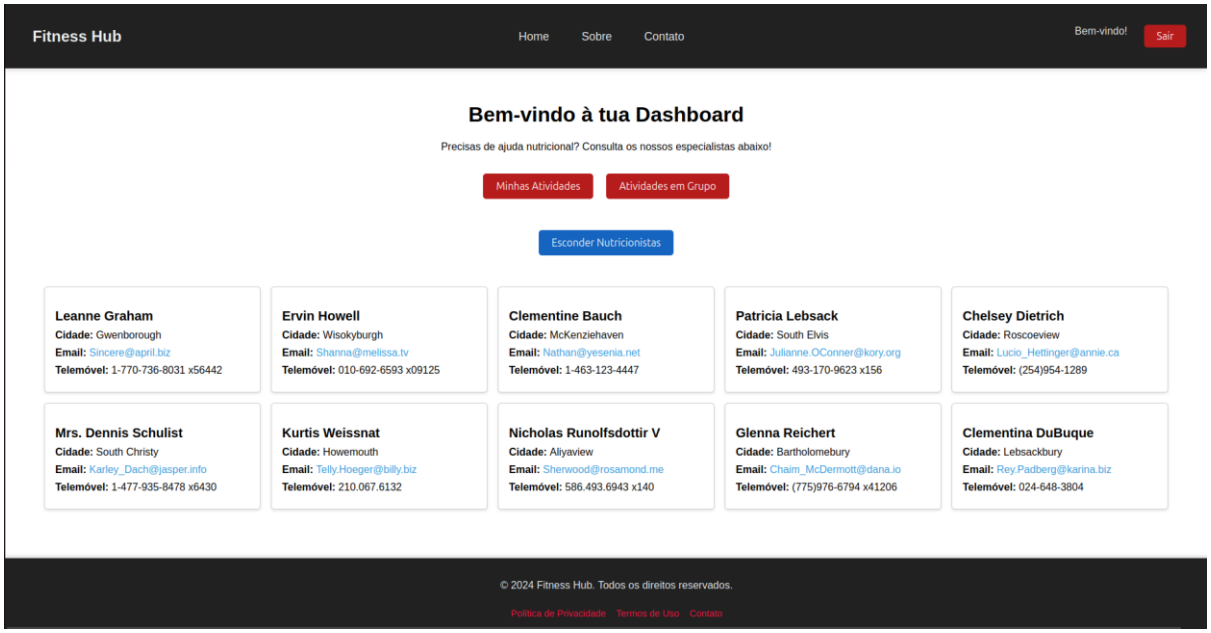

Fig4- DashBoard

Trabalho feito por:
Alexandre Santos 71522,
Pedro Antunes 75409

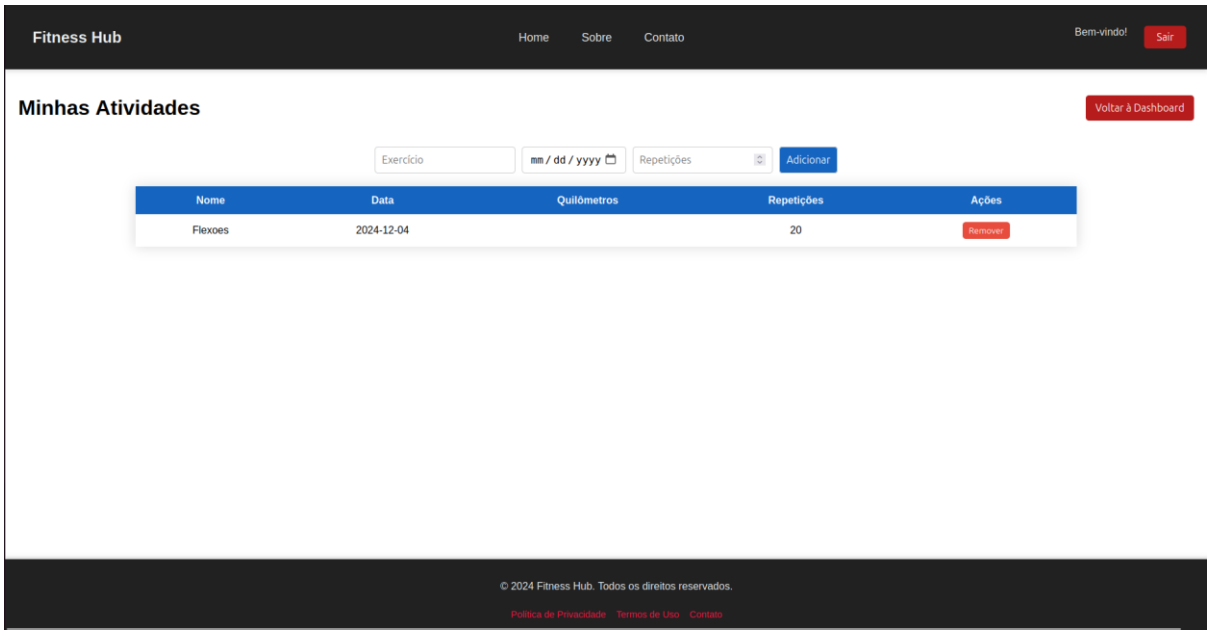

Fig4-Página ”MInhas atividades”

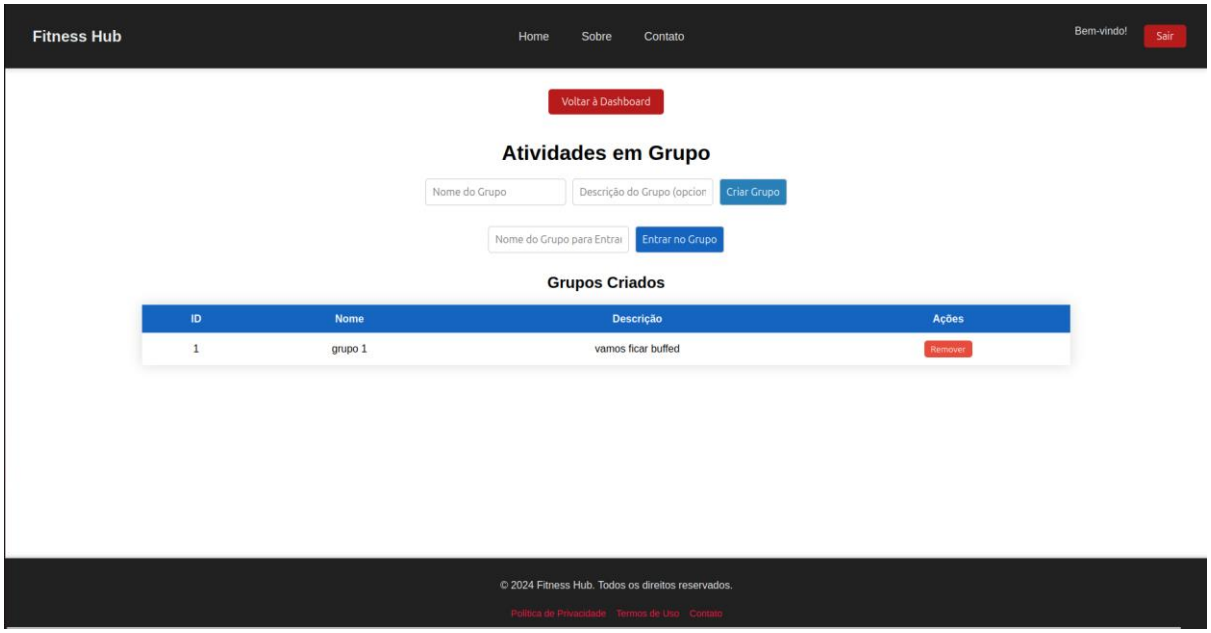

Fig5-Página ”Atividades em grupo”

Rest API, Funcionalidades e requerimentos

Trabalho feito por:
Alexandre Santos 71522,
Pedro Antunes 75409

Neste projeto temos uma pasta chamada routes onde definimos rotas e a criação das tabelas necessárias para a BD (exercicios.ts, grupos.ts, login.ts, register.ts, usuarios.ts), nestes ficheiros também implementamos o POST, GET, DELETE etc.

Também temos o ficheiro server.ts onde irá definir algumas informações do servidor e onde serve como entry point para quando algumas das rotas forem usadas. Neste mesmo ficheiro temos a invocação de um middleware que está definido num ficheiro separado e a implementação de um outro middleware que está neste ficheiro que serve para permitir que os dados json sejam interpretados corretamente e para que se tenha uma comunicação segura e estável através de diferentes origens (isto porque o client usa a porta 8080 e o server 8081).

Para além do middleware mencionado acima também temos outros dois middlewares que são invocados em alguns dos ficheiros presentes na pasta routes e no server.ts, são o errorHandler.ts que é útil para cuidar de erros que possam existir e o validadePassword que irá receber a password do client e verificar se segue as regras necessárias para registar aquela password.

Esta aplicação tem diversas funcionalidades, tais como fazer login, registar-se, dar o seu email como input para receber um email automático de auxílio através da página contato utilizando SMTP, registar exercícios feitos individualmente pelo utilizador ou a criação e inscrição em grupos. Tanto no registo dos exercícios como nos grupos é possível criar, eliminar, etc.

Na página da DashBoard existe um botão chamado “ver nutricionistas” que importa dados de uma API externa fornecida na aula e mostra os nomes e restantes informações lá presentes.

Funcionalidades extras:

Temos algumas funcionalidades e tecnologias diferentes que foram utilizadas. Em vez de usar o NEDB para a BD foi usado o SQLite3 guardando todos os dados no ficheiro usuarios.db.

Para a criação das componentes e views em vez de utilizar o react como nas aulas, decidimos utilizar o vue.js, através disto criámos várias componentes reutilizáveis (como a navbar, footer entre outras) e implementámos as mesmas nas views, onde permite ver o site.

Também implementámos mais alguns middlewares, temos dois que estão em ficheiros separados sendo eles o errorHandler.ts e o validatePassword.ts, o primeiro serve para resposta de erros a inputs dados no client e o segundo serve

Trabalho feito por:
Alexandre Santos 71522,
Pedro Antunes 75409

para a verificação de se a password segue as regras pré-definidas (ter mais que 6 letras, pelo menos um número e um dos seguintes símbolos: !,?,=,#).

Ou seja, neste programa utilizamos de novos middlewares, uma nova framework, uma nova BD e implementamos as outras funcionalidades dadas nas aulas como a utilização de uma API externa entre outras coisas.

Project Set Up

Para dar setup neste programa é simples, apenas é necessário fazer download dos ficheiros do projeto, abrir um terminal na pasta server e escrever “npm install” e de seguida “npm run dev”, depois abrir outro terminal na pasta client e escrever “npm install” e depois “npm start”, ao fazer isto o site será aberto ou terá a opção de o abrir utilizando o localhost através do terminal (segurando com a tecla ctrl e clicando em cima do link localhost a azul).

NOTA: Devido a utilizarmos SQLite e cada membro do grupo utilizar um SO diferente, descobrimos que o ELF Header pode ter incompatibilidades entre o node.js e os binários compilados do package do SQLite3. Se caso depois de executar o comando “npm run dev”, após o npm install, aparecer uma mensagem de erro, quer dizer que o SO da máquina utilizada é diferente do SO da última máquina que atualizou o programa, então para resolver esse problema temos que dar rebuild no sqlite através do comando “npm rebuild sqlite3” e só depois é que é possível fazer o comando “npm run dev” sem dar nenhum erro. Em suma, no caso deste erro acontecer temos que fazer de novo “npm install” depois “npm rebuild sqlite3” e depois novamente “npm run dev”.

Trabalho feito por:
Alexandre Santos 71522,
Pedro Antunes 75409