	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Móveis</p>	<p align="center">Envio de Notificação para o Usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Nessa aula vamos utilizar a biblioteca expo-notifications para enviar mensagens de notificação para o usuário.

<https://docs.expo.dev/versions/latest/sdk/notifications/>

Precisaremos também da biblioteca expo-device para identificar se o dispositivo é capaz de receber notificações.

<https://docs.expo.dev/versions/latest/sdk/device/>

Precisaremos da expo-constants para recuperar o projectId e realizar a requisição.

<https://docs.expo.dev/versions/latest/sdk/constants/>

Toda a explicação do uso está disponível no endereço:

<https://docs.expo.dev/push-notifications/push-notifications-setup/>

Feito isso, vamos preparar o backend para receber o token do celular de quem autenticar no nosso aplicativo, para isso, pegue seu projeto adonisjs e vamos para as configurações.

Inicialmente, precisamos acrescentar uma coluna na tabela de usuários para receber o token do usuário que autenticar. Para isso, vamos criar uma migration que altera a tabela users e acrescenta essa coluna. Para criar a migration execute o comando abaixo.

```
node ace make:migration addColumnTokenToUser
```

O conteúdo do arquivo criado deve ser:

```
TS 1694261624376_add_column_token_to_users.ts ×

database > migrations > TS 1694261624376_add_column_token_to_users.ts > ...
1  import BaseSchema from '@ioc:Adonis/Lucid/Schema'
2
3  export default class extends BaseSchema {
4    protected tableName = 'users'
5
6    public async up() {
7      this.schema.alterTable(this.tableName, (table) => {
8        table.string('token')
9      })
10   }
11
12   public async down() {
13     this.schema.alterTable(this.tableName, (table) => {
14       table.dropColumn('token')
15     })
16   }
17 }
```

Feito isso, vamos criar o controller que a rota deve enviar quando o token for recuperado no APP do usuário. Para criar o controller execute o comando abaixo

```
node ace make:controller User --resource
```

O comando irá criar o controller com todos os métodos mais comuns quando construímos aplicações MVC. Como estamos trabalhando somente API, removi os métodos create e edit que são para Views de mostrar o formulário para inserir e mostrar o formulário para atualizar respectivamente. Vamos trabalhar somente no método update conforme imagem abaixo.


```
TS UsersController.ts M X
app > Controllers > Http > TS UsersController.ts > ...
1  import type { HttpContextContract } from '@ioc:Adonis/Core/HttpContext'
2  import User from 'App/Models/User'
3
4  export default class UsersController {
5    public async index({ }: HttpContextContract) { }
6
7    public async store({ }: HttpContextContract) { }
8
9    public async show({ }: HttpContextContract) { }
10
11    public async update({ request, auth }: HttpContextContract) {
12      const { token } = request.all()
13      const userDB = await User.findOrFail(auth.user?.id)
14      userDB.token = token
15      await userDB.save()
16      return userDB
17    }
18
19    public async destroy({ }: HttpContextContract) { }
20  }
```

Para finalizar a configuração do backend vamos acrescentar a rota que vai receber a requisição e encaminhar para o controller:

```
TS routes.ts M X
start > TS routes.ts > ...
1  import Route from '@ioc:Adonis/Core/Route'
2
3  Route.get('/', async () => {
4    return { hello: 'world' }
5  })
6  Route.post("/register", "AuthController.register")
7  Route.post("/login", "AuthController.login")
8  Route.group(() => {
9    Route.put("/user", "UsersController.update")
10  }).middleware('auth')
```

Observe que a rota é put, ela encaminha a requisição para o método update do UsersController.

Observe também que ela deve ser configurada dentro de um Route.group que tem como tarefa tratar todas as requisições que devem ser autenticadas

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Móveis</p>	<p align="center">Envio de Notificação para o Usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

chamando o middleware auth. Confira se seu middleware está configurado no arquivo start/kernel.ts.

Somente dentro deste group conseguiremos recuperar o id do usuário através do auth.

Feito isso, suba sua aplicação no GitHub e acompanhe o deploy. Caso tenha algum erro, corrija-o antes de prosseguir.

No final da execução, a coluna token deve ser criada na tabela de usuários no Neon, configura se isso realmente ocorreu.

Com o backend configurado, vamos à implementação do aplicativo para que, realizada a autenticação, o token seja enviado ao servidor. Para isso, precisaremos do projectId que é atribuído após configurar o deploy dele no site:

<https://expo.dev/>

Crie uma conta no site caso não tenha, pois iremos fazer, do nosso projeto, o login nessa conta. Para realizar o login execute no terminal.

```
npx expo login
```

Ele deve solicitar login na eas.

```

● MacBook-Pro-de-Lazaro:figma lazaroeuardodasilva$ npx expo login
Log in to EAS
✓ Email or username ... lazarodu
✓ Password ... *****

```

A eas é a ferramenta de configuração para deploy de aplicativos expo, mais informações no endereço.

<https://docs.expo.dev/submit/introduction/>

Para configurar seu projeto execute.

```
npx eas update
```

Responda as perguntas e você verá seu projeto no site expo.dev. Feito isso, seu projeto já tem um ProjectId e conseguiremos configurar o push notifications.

Antes de configurar a função de push, vamos fazer uma alteração na api do usuário. Altere a classe da API acrescentando a função updateToken com o código conforme o abaixo.

```

23  class UserData {
24      register(data: IRegister) {
25          return api.post<IUser>('/register', data);
26      }
27      login(data: IAuthenticate) {
28          return api.post<IUserLogin>('/login', data);
29      }
30      updateToken(token: string) {
31          return api.put('/user', { token })
32      }
33  }

```

Esta função irá consumir a API realizando requisições na rota que atualiza o token no Banco de Dados conforme criamos acima.



Centro Federal de Educação Tecnológica de Minas Gerais
Campus VIII – Varginha
Curso Técnico em Informática

Disciplina
Lab. Aplicações Móveis

Envio de Notificação para o Usuário

Professor
Lázaro Eduardo da Silva

Agora sim, vamos configurar a requisição do token. O código é baseado no endereço:

<https://docs.expo.dev/push-notifications/push-notifications-setup/>

Por uma questão de organização do código criaremos uma função na pasta services/data/Push/index.ts

index.ts U X


src > services > data > Push > index.ts > ...

```
1  import * as Device from 'expo-device';
2  import * as Notifications from 'expo-notifications';
3  import { Alert, Platform } from 'react-native';
4  import Constants from 'expo-constants';
5  import { apiUser } from '../';
6  export interface IExtra {
7    eas: {
8      projectId: string
9    }
10 }
11 export async function registerForPushNotificationsAsync() {
12   let token;
13   if (Platform.OS === 'android') {
14     await Notifications.setNotificationChannelAsync('default', {
15       name: 'default',
16       importance: Notifications.AndroidImportance.MAX,
17       vibrationPattern: [0, 250, 250, 250],
18       lightColor: '#FF231F7C',
19     });
20   }
21
22   if (Device.isDevice) {
23     const { status: existingStatus } = await Notifications.getPermissionsAsync();
24     let finalStatus = existingStatus;
25     if (existingStatus !== 'granted') {
26       const { status } = await Notifications.requestPermissionsAsync()
27       finalStatus = status;
28     }
29     if (finalStatus !== 'granted') {
30       Alert.alert('Falha ao obter o token para envio de mensagens Push');
31     }
32     const extra = Constants.expoConfig?.extra as IExtra
33     token = (await Notifications.getExpoPushTokenAsync({
34       projectId: extra.eas.projectId,
35     }));
36     await apiUser.updateToken(token.data)
37   } else {
38     Alert.alert('Você deve usar um dispositivo físico para receber notificações Push');
39   }
40
41   return token;
42 }
```

A próxima implementação eu coloquei na minha página de Perfil que é a primeira screen acessada após o login. Com isso, minha tela de Perfil ficou com o código:

```
index.tsx M X
src > screens > Perfil > index.tsx > ...
1  import { View, Text } from "react-native";
2  import { styles } from "../styles";
3  import { TouchableOpacity } from "react-native-gesture-handler";
4  import { TabTypes } from "../../navigations/tab.navigation";
5  import * as Notifications from 'expo-notifications';
6  import { registerForPushNotificationsAsync } from "../../services/data/Push";
7  import { useAuth } from "../../hooks/auth";
8  import { useEffect, useState } from "react";
9  import { ComponentLoading } from "../../components";
10 Notifications.setNotificationHandler({
11   handleNotification: async () => ({
12     shouldShowAlert: true,
13     shouldPlaySound: false,
14     shouldSetBadge: false,
15   }),
16 });
17 export function Perfil({ navigation }: TabTypes) {
18   const { user } = useAuth();
19   const [isLoading, setIsLoading] = useState(true);
20   function handleVoltar() {
21     const login = navigation.getParent()
22     login?.goBack()
23   }
24   useEffect(() => {
25     if (user) {
26       setIsLoading(false);
27     }
28   }, [user]);
29   useEffect(() => {
30     async function fetchToken() {
31       const token = await registerForPushNotificationsAsync()
32       console.log(token)
33     }
34     fetchToken()
35   }, []);
36   return (
37     <>
38     {isLoading ? (
39       <ComponentLoading />
40     ) : (
41       <View style={styles.container}>
42         <Text>Perfil</Text>
43         <TouchableOpacity onPress={handleVoltar}>
44           <Text>Voltar</Text>
45         </TouchableOpacity>
46       </View>
47     )}
48   </>
49 )
50 }
```

Observe o código destacado com uma lista azul, que corresponde à alteração e o verde, que corresponde ao código que foi acrescentado.

	<p align="center">Centro Federal de Educação Tecnológica de Minas Gerais Campus VIII – Varginha Curso Técnico em Informática</p>	
<p><i>Disciplina</i> Lab. Aplicações Móveis</p>	<p align="center">Envio de Notificação para o Usuário</p>	<p><i>Professor</i> Lázaro Eduardo da Silva</p>

Feito isso, seu código já é capaz de recuperar o Expo Token do celular do usuário que está usando o seu aplicativo.

Observe que deixamos um console.log na linha 32 para que você identifique este código no terminal. Ele deve ser parecido com o meu:

`ExponentPushToken[_ZlP2fKC80x1Bvh08NrgKz]`

Confira também se este código foi armazenado na tabela de usuários no seu banco de dados.

Para testar, você deve acessar o endereço abaixo.

<https://expo.dev/notifications>

E configurar o Recipient, Message Title, Message Body e clicar em Send a Notification.

A mensagem deve chegar no seu celular, mesmo com o aplicativo fechado.

Bom trabalho!