AULA 08 - Vídeo 6

Introdução

Agora que temos a parte inicial das telas, Vamos começar a fazer a implementação da nossa **Gerência de estados**

Dados que devem ser transmitidos para outras páginas e componentes para outras páginas e componentes, e para isso foi criado a gestão de estados, colocamos nessa biblioteca de gestão, informações que quero que todos componentes do meu sistema consigam consultar de maneira simples

Biblioteca Pinia, o novo padrão do vuex, documentação do pinia
https://pinia.vuejs.org/

Instalaremos usando npm

```
PS C:\Users\Alison Moura\Documents\Curso Vuejs\perfil-profissional-web> npm install pinia

[..................] / idealTree:perfil-profissional-web: sill idealTree buildDeps
```

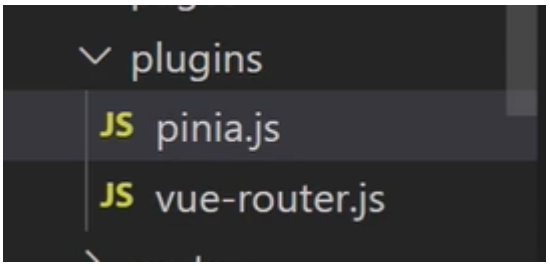
```
PS C:\Users\Alison Moura\Documents\Curso Vuejs\perfil-profissional-web> npm install pinia


added 2 packages, and audited 37 packages in 4s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Alison Moura\Documents\Curso Vuejs\perfil-profissional-web>
```

Criaremos a arquivo na pasta plugins



Em pinia.js

```
import { createPinia } from "pinia";
export default createPinia();
```

main.js

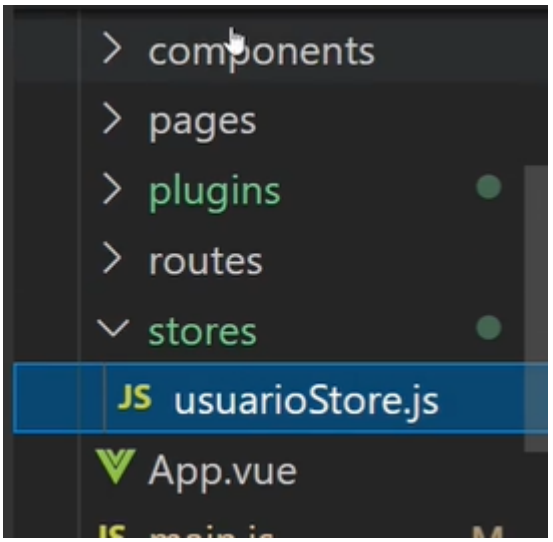
```
import { createApp } from 'vue'
import App from './App.vue'
import router from './plugins/vue-router'
import pinia from "./plugins/pinia";

createApp(App).use(router).use(pinia).mount('#app')
```

Temos agora que fazer um store, um armazenador, um manipulador de estados, e regras de negócios que nao sao de um componente especifico, sao da aplicação inteira
Como listagem de notificações.

Por exemplo, a home vai exibir notificações, assim como a página de notificações também exibirá.
Então esses dados que não são pertencentes a uma única página, se armazena no store do pinia, são os States, e getters que acessam esses estados e manipulam eles e actions que são ações.
Pinia guardam dados que são armazenados globalmente.

Criaremos uma pasta em src



em ‘login.vue’

```
import { mapActions } from 'pinia'
import { useUsuarioStore} from '../stores/usuarioStore'
```

em usuarioStore:

```
import { defineStore } from 'pinia'

export const useUsuarioStore = defineStore('usuario', {
    state: () => ({
        usuarioLogado: {}
    }),
    actions: {
        login() {
            return new Promise((resolve, reject) => {

                console.log("clickou no login");
                setTimeout(() => {  //Comunicação com a API dps
                      this.usuarioLogado = { nome: "jao silva", email:
"jao@gmail.com"}
                resolve();
                }, 500)
            })
           }
        }
})
```

em login.vue novamente

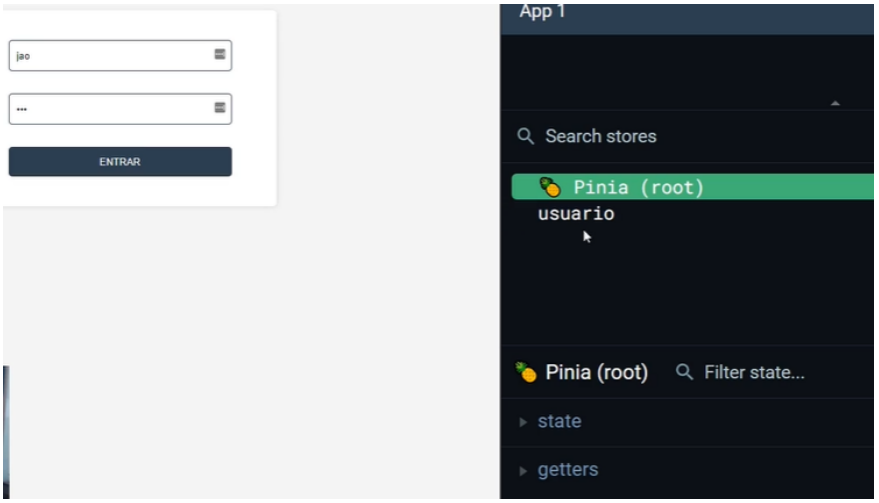
```

export default {
    data: () => ({
        usuario: {}
    }),
    methods:{
            ...mapActions(useUsuarioStore, ["login"]), // vira como um
metodo aqui
        async logar(){
                 await this.login() //mapeamento do mapActions que foi
incorporado
            this.$route.push("/home")
        }
    }
}
</script>
```

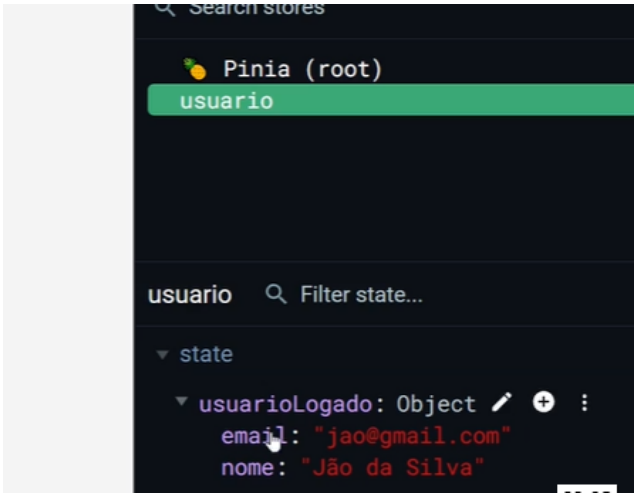
Se a estilização não ficar correta em index.html

```
    <link rel="stylesheet" href="/src/assets/index.css">
```

Para testar vamos ligar novamente o servidor e abrir o vue devtools do navegador e tentar clicar logar.



em state



Corrigindo o 'login.vue'

```
<button type="button" @click="logar">Entrar</button>
```

```
    this.$router.push("/home")
```

Resultado tem que ao clicar redirecionar a rome e o state estar com 'jao' que configuramos

Em 'Home.vue'

```
<template>
    <h1>Home</h1>
    <h2>Ola, {{usuarioLogado.nome}}</h2>
</template>
```

```
<script>
import { mapState } from 'pinia'
import { useUsuarioStore } from '../stores/usuarioStore'

export default {
    computed:{
        ...mapState(useUsuarioStore, ['usuarioLogado'])
    }
}
</script>
```

Resultado:

O pinia nos ajuda a ter dados compartilhados.
E nas próximas aulas faremos conexão com a API.