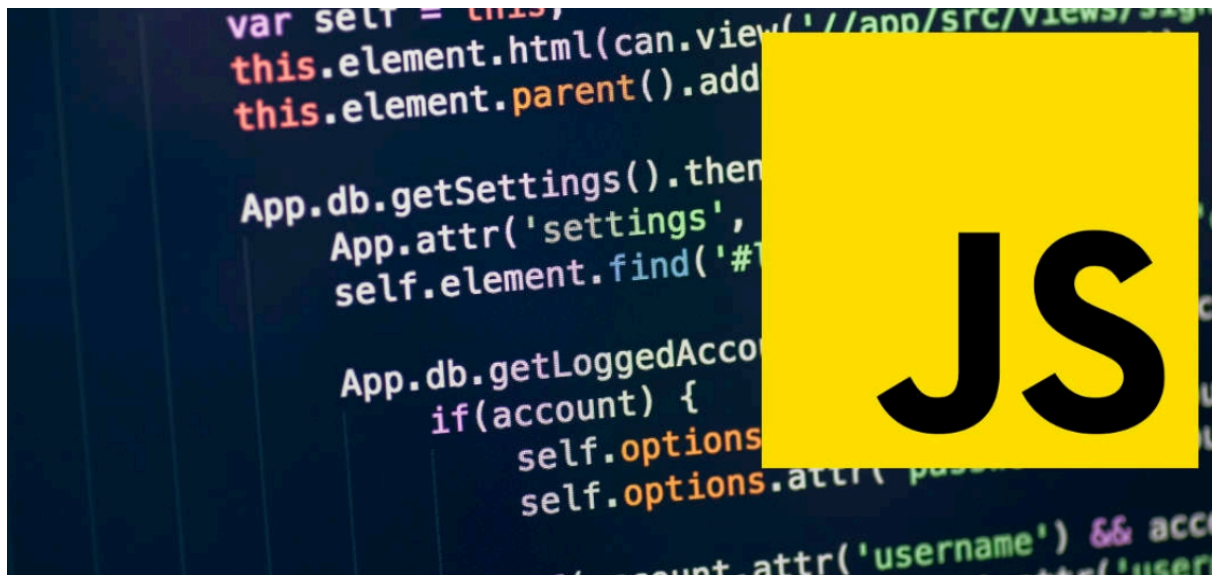


Projecte Javascript



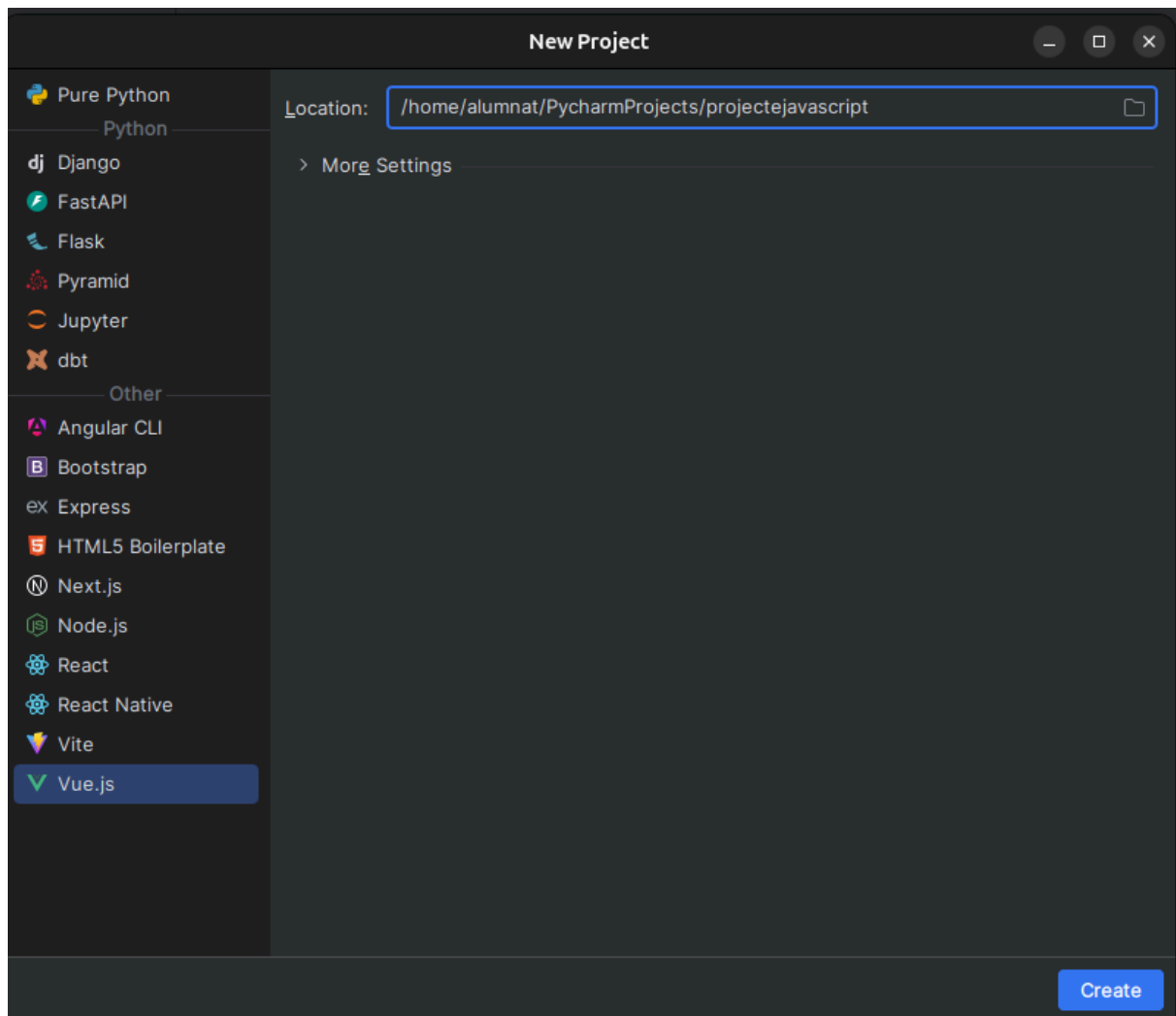
Creat per: Joel Calvet Michavila

Professor: Jordi Vega

Assignatura: MP07, UF2 Projecte Javascript

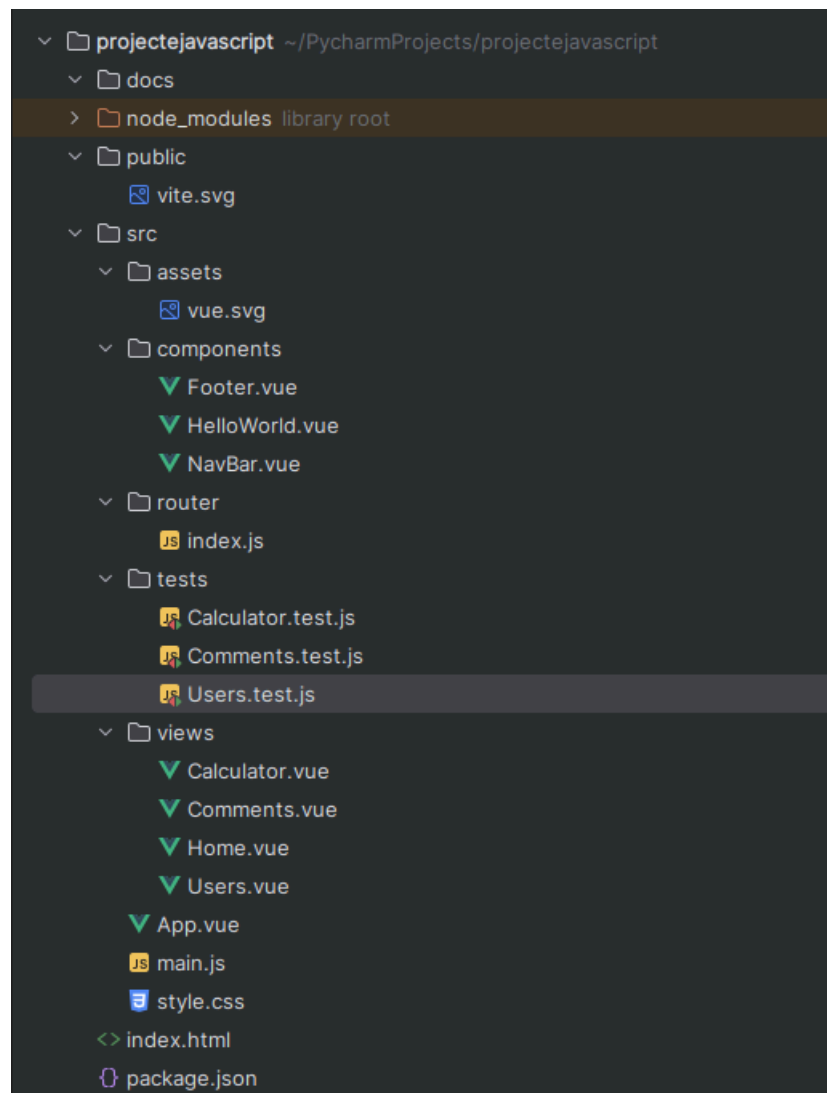
Index

Raons per Fer un Projecte de JavaScript amb Vue.js i PyCharm



Estic fent un projecte de JavaScript amb Vue.js i PyCharm perquè Vue.js és un framework modern i flexible ideal per a aplicacions interactives, i PyCharm és un entorn de desenvolupament robust amb eines avançades per a la depuració, autocompleció de codi, i integració amb sistemes de control de versions com Git. Aquesta combinació em permet crear aplicacions web modulars i eficients amb una gestió de codi millorada.

Estructura del projecte



Descripció de l'estructura de directoris i fitxers

public/

vite.svg: Conté el logotip SVG utilitzat en el projecte.

src/

components/: Conté components reutilitzables de Vue.

NavBar.vue: Component de la barra de navegació.

Footer.vue: Component del peu de pàgina.

views/: Conté les vistes principals de l'aplicació.

Home.vue: Vista principal de l'aplicació amb enllaços a altres funcionalitats.

Users.vue: Vista que mostra una llista d'usuaris obtinguts de l'API DummyJSON.

Comments.vue: Vista que mostra una llista de comentaris obtinguts de l'API JSONPlaceholder.

Calculator.vue: Vista que proporciona una calculadora interactiva.

tests/: Conté les proves unitàries per a les vistes.

Calculator.test.js: Proves unitàries per a la vista de la calculadora.

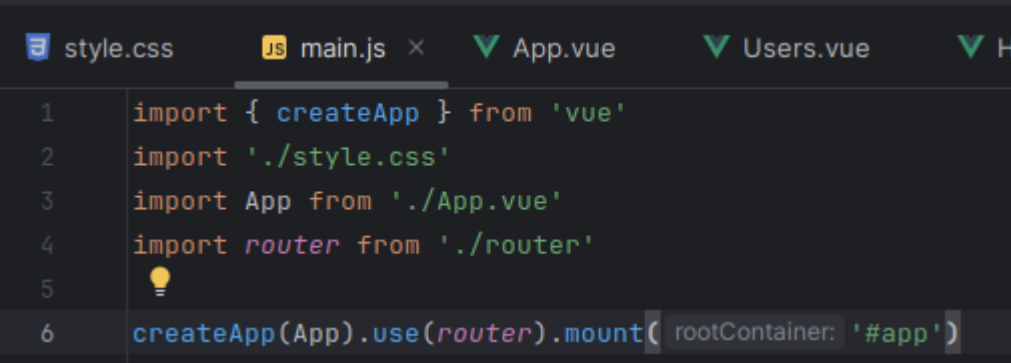
Comments.test.js: Proves unitàries per a la vista de comentaris.

Users.test.js: Proves unitàries per a la vista d'usuaris.

App.vue: Component principal de l'aplicació que inclou la barra de navegació, el contingut principal i el peu de pàgina.

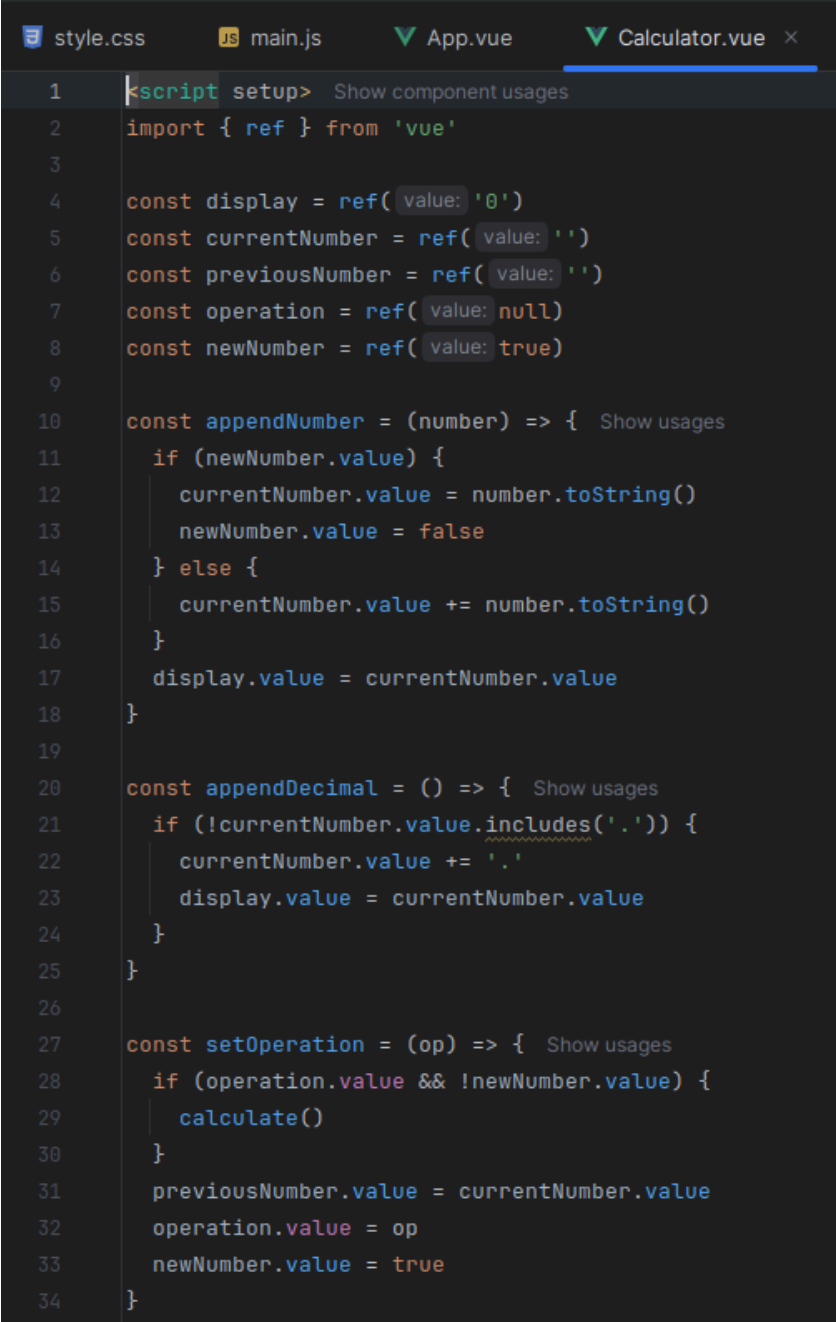
main.js: Punt d'entrada principal de l'aplicació Vue, inicialitza l'aplicació i la munta al DOM.

style.css: Fitxer CSS que inclou les directrius de Tailwind CSS i algunes regles de estil globals.



```
1 import { createApp } from 'vue'
2 import './style.css'
3 import App from './App.vue'
4 import router from './router'
5
6 createApp(App).use(router).mount({ rootContainer: '#app'})
```

Aquest codi inicialitza una aplicació Vue 3. Primer, importa la funció createApp de Vue, que permet crear la instància de l'aplicació. A continuació, importa el fitxer d'estil global (style.css) i el component principal (App.vue), que és el punt d'entrada de l'aplicació. També importa el sistema de rutes (router), que gestiona la navegació entre pàgines. Finalment, createApp(App).use(router).mount('#app') crea l'aplicació, li afegeix el sistema de rutes i la "munta" a l'element HTML amb l'ID #app, de manera que es pugui renderitzar a la pàgina web.



```
1  <script setup> Show component usages
2  import { ref } from 'vue'
3
4  const display = ref( value: '0')
5  const currentNumber = ref( value: '')
6  const previousNumber = ref( value: '')
7  const operation = ref( value: null)
8  const newNumber = ref( value: true)
9
10 const appendNumber = (number) => { Show usages
11   if (newNumber.value) {
12     currentNumber.value = number.toString()
13     newNumber.value = false
14   } else {
15     currentNumber.value += number.toString()
16   }
17   display.value = currentNumber.value
18 }
19
20 const appendDecimal = () => { Show usages
21   if (!currentNumber.value.includes('.')) {
22     currentNumber.value += '.'
23     display.value = currentNumber.value
24   }
25 }
26
27 const setOperation = (op) => { Show usages
28   if (operation.value && !newNumber.value) {
29     calculate()
30   }
31   previousNumber.value = currentNumber.value
32   operation.value = op
33   newNumber.value = true
34 }
```

El codi configura la lògica bàsica d'una calculadora en Vue 3 utilitzant Composition API. Defineix referències reactives per a la visualització (display), el número actual (currentNumber), el número anterior (previousNumber), l'operació (operation) i una bandera per saber si es tracta d'un número nou (newNumber). Les funcions implementades són: appendNumber, que afegeix dígitos al número actual i actualitza el display; appendDecimal, que incorpora el punt decimal si encara no hi és; i setOperation, que guarda el número actual com a previousNumber, estableix l'operació i prepara el sistema per a l'entrada del pròxim número.

```

35
36 const calculate = () => { Show usages
37   if (!operation.value || newNumber.value) return
38
39   const prev = parseFloat(previousNumber.value)
40   const current = parseFloat(currentNumber.value)
41   let result
42
43   switch (operation.value) {
44     case '+':
45       result = prev + current
46       break
47     case '-':
48       result = prev - current
49       break
50     case '*':
51       result = prev * current
52       break
53     case '/':
54       result = prev / current
55       break
56     default:
57       return
58   }
59
60   currentNumber.value = result.toString()
61   display.value = currentNumber.value
62   operation.value = null
63   newNumber.value = true
64 }

```

Aquest bloc afegeix la funció `calculate`, encarregada de realitzar operacions aritmètiques entre dos números (`previousNumber` i `currentNumber`). Primer, comprova si hi ha una operació pendent i si `newNumber` no és cert, evitant així càlculs innecessaris. Converteix els valors emmagatzemats a nombres de punt flotant (`prev` i `current`). Després, segons el valor de `operation`, executa una de les quatre operacions: suma (+), resta (-), multiplicació (*) o divisió (/), emmagatzemant el resultat a `result`. Aquest resultat servirà per actualitzar la interfície de la calculadora.

```

</button>
<button @click="() => setOperation(op: '/')"
      class="h-16 bg-gradient-to-br from-violet-600 to-violet-800 text-white rounded-2xl hover:from-violet-700 hover:to-violet-900 trans
      ÷
</button>
<button @click="() => setOperation(op: '*')"
      class="h-16 bg-gradient-to-br from-violet-600 to-violet-800 text-white rounded-2xl hover:from-violet-700 hover:to-violet-900 trans
      ×
</button>

<!-- Números 7-9 -->
<button v-for="n in [7,8,9]" :key="n" @click="() => appendNumber(n)"
      class="h-16 bg-gradient-to-br from-gray-700 to-gray-800 text-white rounded-2xl hover:from-gray-600 hover:to-gray-700 transform hov
      {{ n }}
</button>
<button @click="() => setOperation(op: '-')"
      class="h-16 bg-gradient-to-br from-violet-600 to-violet-800 text-white rounded-2xl hover:from-violet-700 hover:to-violet-900 trans
      -
</button>

```

Aquest bloc defineix els botons de la interfície d'una calculadora amb estil i funcionalitat per a les operacions bàsiques i els números. Cada botó d'operació (\div , \times , $-$) crida la funció `setOperation` amb el símbol corresponent per establir l'operació seleccionada. Els botons de números (7, 8, 9) es generen amb `v-for`, i cadascun executa la funció `appendNumber(n)` per afegir el número al valor actual de la calculadora. L'estil inclou degradats violeta per a operacions i grisos per a números, amb efectes d'escala i transició que milloren l'interactivitat visual. Aquests elements proporcionen una experiència atractiva i consistent per a l'usuari.


```

1  <script setup> Show component usages
2  import { ref, onMounted } from 'vue'
3
4  const comments = ref( value: [])
5  const loading = ref( value: true)
6  const error = ref( value: null)
7
8  onMounted( hook: async () => {
9    try {
10      const response = await fetch( input: 'https://jsonplaceholder.typicode.com/comments')
11      if (!response.ok) throw new Error('Error al carregar els comentaris')
12      comments.value = await response.json()
13    } catch (e) {
14      error.value = e.message
15    } finally {
16      loading.value = false
17    }
18  })
19 </script>

```

Aquest codi configura la càrrega asincrònica de comentaris des d'una API utilitzant Vue 3 amb la funció `onMounted` i la Composition API. Es defineixen tres referències reactivament: `comments` per emmagatzemar la llista de comentaris, `loading` per indicar si els comentaris s'estan carregant, i `error` per capturar qualsevol error durant la càrrega. Quan el component es munta (`onMounted`), s'executa una crida a l'API `fetch` per obtenir els comentaris des de `https://jsonplaceholder.typicode.com/comments`. Si la resposta és vàlida, els comentaris es guarden a `comments.value`. Si hi ha un error, es captura i s'emmagatzema el missatge d'error a `error.value`. Finalment, es marca el procés de càrrega com a complet (`loading.value = false`).

```

<template>
  <div class="container mx-auto px-4 py-8">
    <h1 class="text-3xl font-bold mb-6">Comentaris</h1>

    <div v-if="loading" class="text-center">
      <p class="text-xl">Carregant comentaris...</p>
    </div>

    <div v-else-if="error" class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded">
      {{ error }}
    </div>

    <div v-else class="grid gap-4">
      <div v-for="comment in comments" :key="comment.id" class="bg-white p-4 rounded-lg shadow">
        <h2 class="text-xl font-semibold mb-2">{{ comment.name }}</h2>
        <p class="text-gray-600 mb-2">{{ comment.email }}</p>
        <p>{{ comment.body }}</p>
      </div>
    </div>
  </div>
</template>

```

Aquest codi defineix una plantilla en Vue per mostrar comentaris obtinguts d'una API amb Tailwind CSS per al disseny. La pàgina té tres possibles estats: mentre es carreguen els comentaris, es mostra un missatge de càrrega; si hi ha un error, es mostra un missatge d'error amb un fons vermell; i, un cop carregats els comentaris, es renderitza una llista de comentaris dins de targetes blanques amb ombres i espaiats adequats. Les dades de cada comentari (nom, correu electrònic i cos del comentari) es mostren de manera clara i organitzada.

```

1  <script setup> Show component usages
2  import { ref, onMounted } from 'vue'
3  import axios from 'axios'
4
5  const users = ref( value: [])
6  const loading = ref( value: true)
7  const error = ref( value: null)
8
9  onMounted( hook: async () => {
10     try {
11       const response = await axios.get( url: 'https://dummyjson.com/users')
12       users.value = response.data.users
13     } catch (e) {
14       error.value = e.message
15     } finally {
16       loading.value = false
17     }
18   })
19 </script>
20

```

Aquest codi utilitza Vue 3 i la biblioteca axios per realitzar una crida HTTP a l'API <https://dummyjson.com/users> per obtenir una llista d'usuaris. Quan el component es munta (onMounted), es fa una petició asincrònica amb axios.get, i els usuaris es guarden a la referència reactiva users. Si la petició té èxit, els usuaris es guarden en users.value. Si es produeix un error, es captura el missatge d'error i es guarda a error.value. Finalment, independentment del resultat, es marca la variable loading com a false per indicar que la càrrega ha finalitzat.

```

<template>
  <div class="container mx-auto px-4 py-8">
    <h1 class="text-3xl font-bold mb-6">Usuaris</h1>

    <div v-if="loading" class="text-center">
      <p class="text-xl">Carregant usuaris...</p>
    </div>

    <div v-else-if="error" class="bg-red-100 border border-red-400 text-red-700 px-4 py-3 rounded">
      {{ error }}
    </div>

    <div v-else class="grid md:grid-cols-2 lg:grid-cols-3 gap-4">
      <div v-for="user in users" :key="user.id" class="bg-white p-4 rounded-lg shadow">
        
        <h2 class="text-xl font-semibold text-center mb-2">{{ user.firstName }} {{ user.lastName }}</h2>
        <div class="text-gray-600">
          <p><strong>Correu:</strong> {{ user.email }}</p>
          <p><strong>Telèfon:</strong> {{ user.phone }}</p>
          <p><strong>Empresa:</strong> {{ user.company.name }}</p>
          <p><strong>Adreça:</strong> {{ user.address.city }}, {{ user.address.state }}</p>
        </div>
      </div>
    </div>
  </div>
</template>

```

Aquest codi crea una pàgina per mostrar una llista d'usuaris amb Vue 3 i Tailwind CSS. La interfície es divideix en tres estats: si els usuaris s'estan carregant, es mostra un missatge de càrrega; si hi ha un error durant la càrrega, es mostra un missatge d'error amb un fons vermell; i si els usuaris es carreguen correctament, es mostra una graella responsiva amb informació com el nom, correu electrònic, telèfon, empresa i adreça de cada usuari. Les imatges dels usuaris es mostren en un cercle i el disseny s'adapta a dispositius de diferents mides, utilitzant classes de Tailwind per a una disposició en múltiples columnes.

```

style.css  main.js  App.vue  Footer.vue  Calculator.vue  Users.vue  Hor
1  <template> Show component usages
2  <footer class="bg-gradient-to-r from-purple-800 to-indigo-900 text-white py-4 mt-auto">
3    <div class="container mx-auto text-center">
4      <p>© 2024 Projecte Vue SPA. Creat per Joel Calvet</p>
5    </div>
6  </footer>
7  </template>

```

Aquesta és la plantilla per al footer.

```
style.css main.js App.vue Footer.vue HelloWorld.vue NavBar.vue Calculator.vue Users.vue Home.vue C
1 <script setup> Show component usages
2 import { ref } from 'vue'
3
4 const isOpen = ref( value: false)
5 </script>
6
7 <template>
8   <nav class="bg-gradient-to-r from-purple-800 to-indigo-900">
9     <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
10       <div class="flex items-center justify-between h-16">
11         <div class="flex items-center">
12           <div class="flex-shrink-0">
13             <router-link to="/" class="text-white font-bold">Vue SPA</router-link>
14           </div>
15           <div class="hidden md:block">
16             <div class="ml-10 flex items-baseline space-x-4">
17               <router-link to="/" class="text-gray-300 hover:text-white px-3 py-2 rounded-md text-sm font-medium">Inici</router-link>
18               <router-link to="/calculator" class="text-gray-300 hover:text-white px-3 py-2 rounded-md text-sm font-medium">Calculadora</router-link>
19               <router-link to="/comments" class="text-gray-300 hover:text-white px-3 py-2 rounded-md text-sm font-medium">Comentaris</router-link>
20               <router-link to="/users" class="text-gray-300 hover:text-white px-3 py-2 rounded-md text-sm font-medium">Usuaris</router-link>
21             </div>
22           </div>
23         </div>
24         <div class="md:hidden">
25           <button @click="isOpen = !isOpen" class="text-gray-300 hover:text-white">
26             <svg class="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor">
27               <path v-if="!isOpen" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
28               <path v-else stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6L12 12" />
29             </svg>
30           </button>
31         </div>
32       </div>
33     </div>
34   </nav>
```

Aquest codi crea una barra de navegació (navbar) en una aplicació Vue amb un disseny adaptatiu, utilitzant Tailwind CSS per estilitzar-la. La barra de navegació té un fons amb un gradient de colors (from-purple-800 to-indigo-900). Conté un enllaç al nom de l'aplicació (Vue SPA) que redirigeix a la pàgina d'inici i diversos enllaços de navegació (usant router-link) per accedir a diferents seccions de l'aplicació, com la pàgina d'inici, la calculadora, els comentaris i els usuaris. En dispositius més grans (mida md o superior), els enllaços es mostren horitzontalment; mentre que en dispositius més petits, un botó de menú (button) es fa visible per permetre l'obertura i tancament d'un menú de navegació a la pantalla (aquest comportament es controla mitjançant la variable reactiva isOpen). El botó de menú alterna entre un icono de "menú" i un de "tancar", depenent de l'estat de isOpen.

```

<div class="md:hidden">
  <button @click="isOpen = !isOpen" class="text-gray-300 hover:text-white">
    <svg class="h-6 w-6" fill="none" viewBox="0 0 24 24" stroke="currentColor">
      <path v-if="!isOpen" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />
      <path v-else stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6L12 12" />
    </svg>
  </button>
</div>
</div>
</div>
<div v-if="isOpen" class="md:hidden">
  <div class="px-2 pt-2 pb-3 space-y-1 sm:px-3">
    <router-link to="/" class="text-gray-300 hover:text-white block px-3 py-2 rounded-md text-base font-medium">Inici</router-link>
    <router-link to="/calculator" class="text-gray-300 hover:text-white block px-3 py-2 rounded-md text-base font-medium">Calculadora</router-link>
    <router-link to="/comments" class="text-gray-300 hover:text-white block px-3 py-2 rounded-md text-base font-medium">Comentaris</router-link>
    <router-link to="/users" class="text-gray-300 hover:text-white block px-3 py-2 rounded-md text-base font-medium">Usuaris</router-link>
  </div>
</div>
</nav>
</template>

```

Aquest codi completa la barra de navegació per a dispositius mòbils. Quan la vista es redueix (mitjançant la classe `md:hidden` de Tailwind), un botó es mostra per alternar la visibilitat del menú mitjançant l'ús de la variable reactiva `isOpen`. En fer clic sobre aquest botó, el valor de `isOpen` canvia, i el botó canvia la seva icona entre un símbol de "menú" (tres línies horitzontals) i un de "tancar" (una creu) en funció de l'estat del menú. Quan `isOpen` és `true`, es mostra un menú vertical amb enllaços a les seccions de l'aplicació: "Inici", "Calculadora", "Comentaris" i "Usuaris", que són components de Vue accessibles mitjançant `router-link`. Cada enllaç ocupa tot l'espai disponible per garantir una bona experiència en dispositius tàctils, amb un disseny net i clar gràcies a les classes de Tailwind.

Benvinguts al Projecte Vue SPA

Una aplicació completa d'una sola pàgina construïda amb Vue 3

Calculadora

Realitza operacions matemàtiques amb la nostra calculadora interactiva

[Prova la Calculadora](#)

Comentaris

Visualitza comentaris obtinguts de l'API JSONPlaceholder

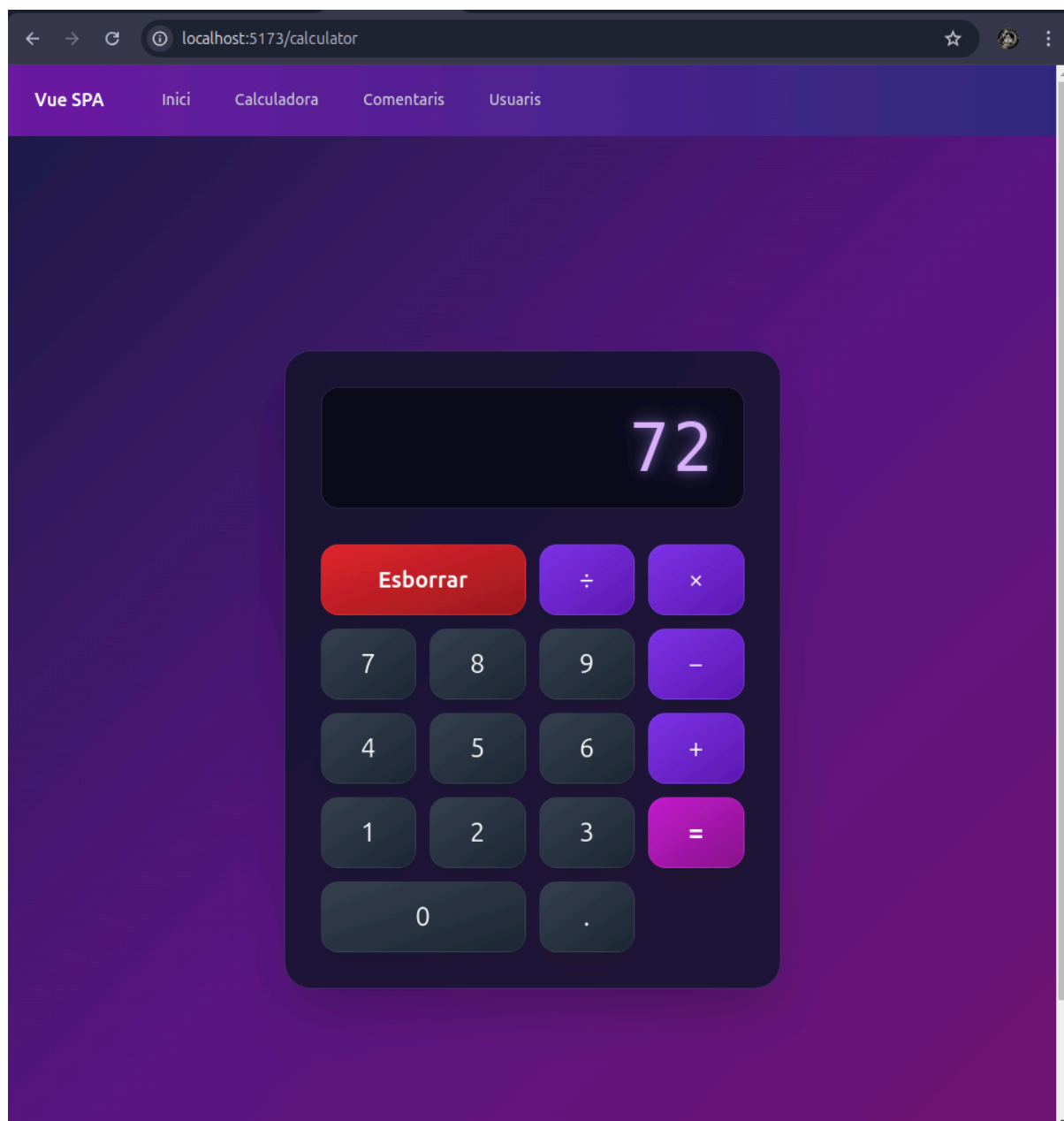
[Veure Comentaris](#)

Usuaris

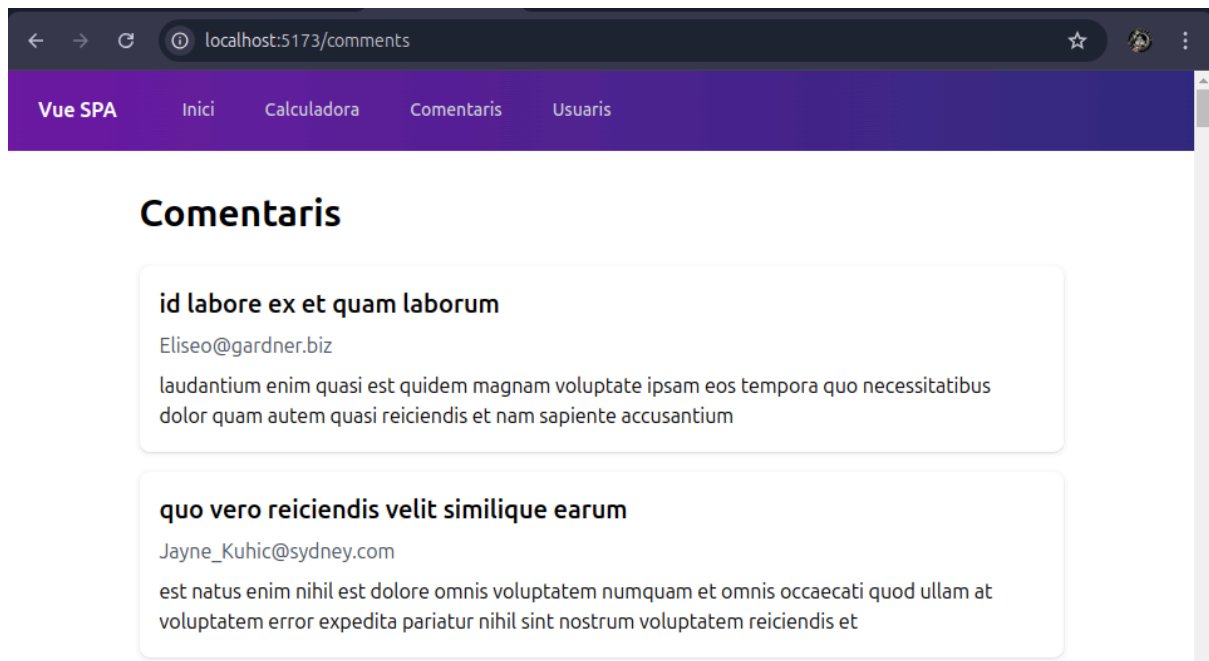
Explora les dades d'usuaris de l'API DummyJSON

[Veure Usuaris](#)

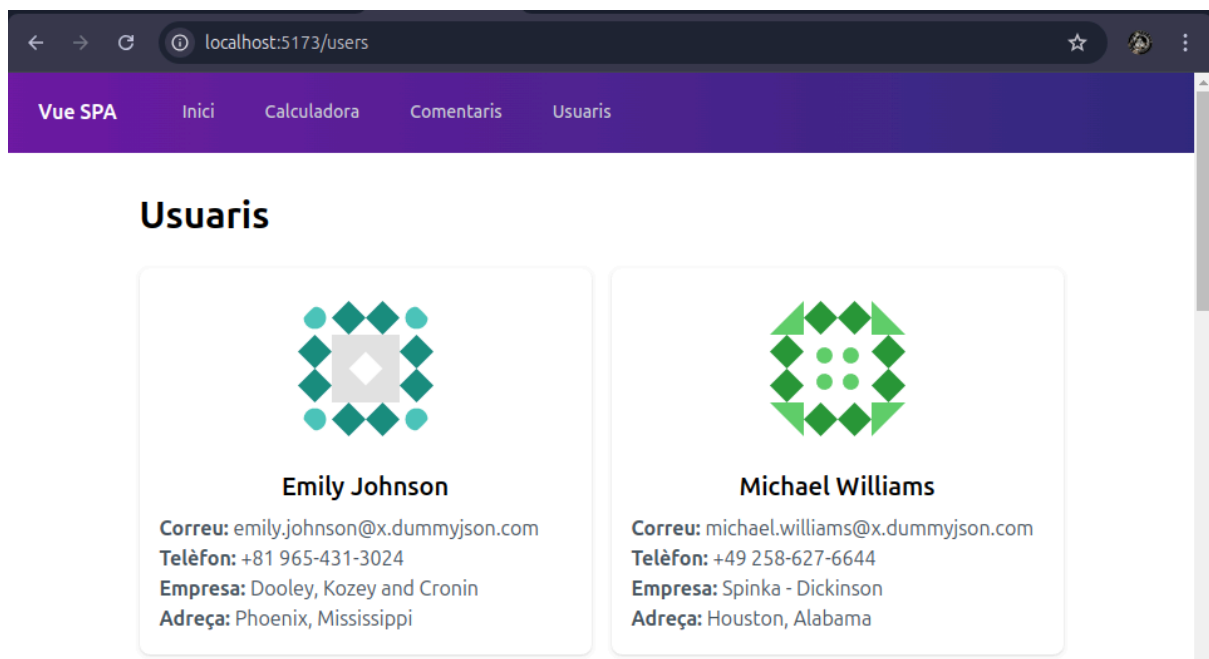
Aquesta és la pàgina principal.



Així es veu la calculadora.



Aquesta és la pàgina de comentaris.



I aquesta la d'usuaris.

Tests

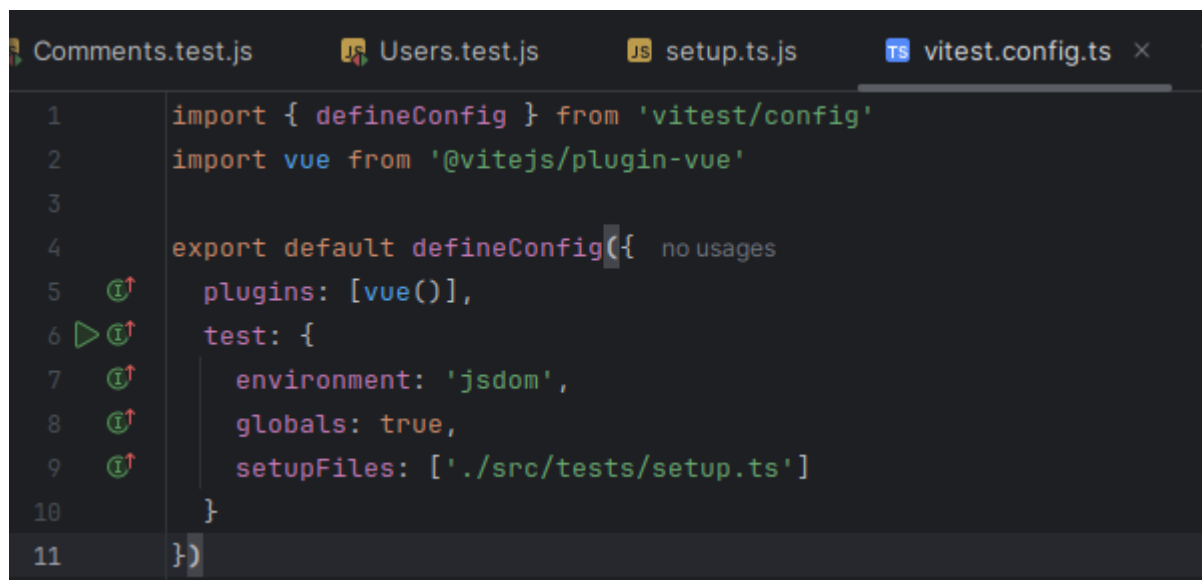
```
→ projectejavascript npm install --save-dev vitest @vue/test-utils

up to date, audited 229 packages in 3s

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

found 0 vulnerabilities
```

Amb aquesta comanda instal·lo el necessari per a fer els tests.



```
Comments.test.js  Users.test.js  setup.ts.js  vitest.config.ts x
1  import { defineConfig } from 'vitest/config'
2  import vue from '@vitejs/plugin-vue'
3
4  export default defineConfig({ no usages
5    plugins: [vue()],
6    test: {
7      environment: 'jsdom',
8      globals: true,
9      setupFiles: ['./src/tests/setup.ts']
10   }
11 })
```

Aquest fitxer de configuració per a Vitest amb Vite i Vue defineix un entorn de test basat en jsdom, que simula un navegador per executar els tests en un entorn de JavaScript. La configuració utilitza el plugin @vitejs/plugin-vue per permetre la compilació de components Vue i estableix un fitxer de configuració addicional (setup.ts) per inicialitzar qualsevol configuració prèvia, com mocks globals o spies. Els tests poden aprofitar aquest entorn per verificar la funcionalitat dels components Vue, incloent la interacció amb les API com fetch o l'ús de biblioteques externes, simulant un comportament similar al del navegador real.

```
Calculator.test.js
1 import { mount } from '@vue/test-utils'
2 import { describe, it, expect } from 'vitest'
3 import Calculator from '../views/Calculator.vue'
4
5 describe('Calculator.vue', () :void => {
6   it('renders calculator component', () :void => {
7     const wrapper : VueWrapper<..., ComponentPublicInstance<...> > = mount(Calculator)
8     expect(wrapper.exists()).toBe( expected: true)
9   })
10
11   it('has clear button', () :void => {
12     const wrapper : VueWrapper<..., ComponentPublicInstance<...> > = mount(Calculator)
13     const clearButton : DOMWrapper<HTMLButtonElement> = wrapper.find( selector: 'button')
14     expect(clearButton.text()).toContain( item: 'Esborrar')
15   })
16 })
```

Aquest codi és un conjunt de tests per al component Calculator.vue utilitzant Vitest i Vue Test Utils. El primer test comprova si el component es renderitza correctament, assegurant-se que el component existeix en el DOM després de ser muntat. El segon test verifica si el component inclou un botó d'esborrar que mostra el text "Esborrar". S'utilitza mount per muntar el component en un entorn de test, i després es busca el botó amb wrapper.find('button') per verificar que el text mostrat sigui el correcte. Els tests estan agrupats sota un describe per proporcionar un conjunt lògic d'especificacions per al component.

```
lock.json  <> index.html  Calculator.test.js  Comments.test.js  Users.test.js  JS se
1  > import ...
4
5  >> describe('Comments.vue', () :void => {
6  >   it('renders comments component', () :void => {
7     const wrapper : VueWrapper<...>, ComponentPublicInstance<...> = mount(Comments)
8     expect(wrapper.exists()).toBe( expected: true)
9   })
10
11 >   it('shows loading state initially', () :void => {
12     const wrapper : VueWrapper<...>, ComponentPublicInstance<...> = mount(Comments)
13     expect(wrapper.text()).toContain( item: 'Carregant')
14   })
15 }]
```

Aquest codi defineix dos tests per al component Comments.vue utilitzant Vitest i Vue Test Utils. El primer test comprova si el component es renderitza correctament, verificant que el component existeix en el DOM després de ser muntat. El segon test verifica que el component mostri un missatge de càrrega inicialment, comprovant que el text "Carregant" apareix en el DOM quan els comentaris encara s'estan carregant. Els tests es realitzen mitjançant mount, que munta el component completament, i wrapper.text(), que s'utilitza per verificar el contingut textual del component. Els tests estan agrupats sota un describe per organitzar les especificacions del component.

```
lock.json  <> index.html  Calculator.test.js  Comments.test.js  Users.test.js  JS se
1  > import ...
4
5  describe('Comments.vue', () :void => {
6    it('renders comments component', () :void => {
7      const wrapper : VueWrapper<..., ComponentPublicInstance<...> = mount(Comments)
8      expect(wrapper.exists()).toBe( expected: true)
9    })
10
11   it('shows loading state initially', () :void => {
12     const wrapper : VueWrapper<..., ComponentPublicInstance<...> = mount(Comments)
13     expect(wrapper.text()).toContain( item: 'Carregant')
14   })
15 })
```

Aquest codi conté una sèrie de tests per al component `Comments.vue`, utilitzant Vitest i Vue Test Utils. El primer test verifica que el component `Comments` es renderit correctament al muntar-se, assegurant-se que el component existeix al DOM després d'executar-se. El segon test comprova que el component mostri un estat de càrrega inicialment. Aquest test comprova si el missatge de "Carregant" apareix en el DOM quan els comentaris encara no s'han carregat. Els tests utilitzen el mètode `mount` per muntar el component i `wrapper.text()` per verificar el contingut textual dins del component. Tot això es troba agrupat dins de la funció `describe` per organitzar millor les especificacions del component.

```
lock.json  <> index.html  Calculator.test.js  Comments.test.js  Users.test.js  JS se
1  import { beforeAll, afterAll } from 'vitest'
2
3  beforeAll( fn: () :void => {
4    // Code to run before all tests
5    console.log('Setting up tests...')
6  })
7
8  afterAll( fn: () :void => {
9    // Code to run after all tests
10   console.log('Cleaning up after tests...')
11 })
```

Aquest codi utilitza Vitest per configurar operacions que es realitzaran abans i després de l'execució dels tests. La funció `beforeAll()` s'executa abans de començar qualsevol test, en aquest cas, es mostra un missatge a la consola amb el text "Setting up tests...", el qual es podria utilitzar per fer inicialitzacions globals o mocks que es necessiten per als tests. D'altra banda, la funció `afterAll()` s'executa després que tots els tests s'hagin completat, i en aquest cas, es mostra el missatge "Cleaning up after tests...", el qual es podria utilitzar per netejar recursos o restablir l'estat de l'entorn de test. Aquestes funcions permeten controlar l'entorn de test abans i després de l'execució dels tests.

```
DEV v1.6.0 /home/alumnat/PycharmProjects/projectejavascript

stdout | src/tests/Calculator.test.js
Setting up tests...
Cleaning up after tests...

stdout | src/tests/Users.test.js
Setting up tests...
Cleaning up after tests...

stdout | src/tests/Comments.test.js
Setting up tests...
Cleaning up after tests...

✓ src/tests/Users.test.js (2)
✓ src/tests/Comments.test.js (2)
✓ src/tests/Calculator.test.js (2)

Test Files  3 passed (3)
  Tests     6 passed (6)
Start at    16:47:56
Duration    985ms (transform 247ms, setup 18ms, collect 757ms, tests 152ms, environment 975ms, prepare 520ms)

PASS Waiting for file changes...
      press h to show help, press q to quit
```

Els tests han sigut satisfactoris.