Fecha: Diciembre 02 del 2024

Docente: Alexander Toscano

Presentado por: DANILO AGUILAR, CRISTIAN GALLEGO , RADYS BALLESTEROS

DOCUMENTACION

Pinia

Pinia es una librería de gestión de estado utilizada en aplicaciones modernas de Vue.js, que reemplaza a Vuex. En este proyecto, Pinia se usa para centralizar y gestionar los datos de los eventos en toda la aplicación.

- **State**: Es el lugar donde se almacena la información reactiva de la aplicación, como la lista de eventos y un solo evento.
- **Getters**: Son propiedades computadas que permiten obtener datos del estado, como obtener todos los eventos o un evento específico.
- **Actions**: Son funciones que permiten modificar el estado o realizar operaciones asincrónicas (como obtener datos de un servidor). En este proyecto, `fetchEvents` es la acción encargada de traer los datos de los eventos.

Pinia ayuda a gestionar el estado de forma predecible a lo largo de la aplicación, lo que hace que la aplicación sea más escalable y fácil de mantener.

Store

El **store** (ubicado en `products.ts`) contiene la lógica para gestionar los eventos en la aplicación. Está estructurado como un **Pinia store** donde se define el estado de los eventos y se implementan las acciones (como la de obtener datos). La propiedad `persist` asegura que el estado se guarde en el almacenamiento local del navegador, por lo que incluso si el usuario recarga la página, el estado se mantiene consistente.

En este caso, el store maneja el estado de los eventos y facilita su acceso desde cualquier componente de la aplicación, manteniendo los datos sincronizados.

Server

El **servidor** es responsable de manejar las solicitudes de la API y proporcionar los datos necesarios al frontend. En este proyecto, el servidor utiliza un controlador del lado del

servidor (`index.get.ts`) para devolver los datos de los eventos cuando se realiza una solicitud GET al endpoint `/api/events`.

El servidor no realiza operaciones complejas, simplemente responde a las solicitudes de eventos con datos predefinidos en formato JSON, que luego serán utilizados por el frontend.

Componentes

Los **componentes** son unidades reutilizables de la interfaz de usuario en Vue.js. En este proyecto, cada componente probablemente es responsable de renderizar una parte específica de la UI, como una lista de eventos, los detalles de un evento específico, o formularios para agregar o editar eventos.

Los componentes interactúan con el **store de Pinia** para obtener los datos de los eventos y mostrarlos dinámicamente. Esto permite que los datos de los eventos sean accesibles y manipulables desde cualquier parte de la aplicación sin duplicar la lógica de gestión del estado.

Componentes del Proyecto

En la carpeta **`components`**, encontramos los siguientes archivos:

1. **`Calendar.vue`**:
    - Este componente probablemente se encarga de representar un calendario visual, donde los usuarios pueden ver los eventos programados. En aplicaciones de gestión de eventos, un calendario es esencial para organizar y visualizar los eventos por fecha.
2. **`Event.vue`**:
    - Este componente probablemente muestra la información detallada de un solo evento. Podría incluir el título, la descripción, la fecha y otros detalles específicos de un evento en particular.
3. **`MenuMeses.vue`**:
    - Es posible que este componente se utilice para permitir a los usuarios navegar entre diferentes meses en el calendario. Se puede usar para filtrar o ver los eventos de un mes específico.
4. **`Meses.vue`**:
    - Este componente podría estar relacionado con la visualización de los meses del año o con la presentación de eventos de un mes en particular. Tal vez actúa como una barra de navegación entre meses o como un selector.
5. **`Proximos.vue`**:
    - Este componente parece estar destinado a mostrar los eventos más cercanos o próximos. Podría ser útil para que los usuarios vean los próximos eventos sin necesidad de desplazarse por todo el calendario.

Páginas del Proyecto

En la carpeta **`pages`**, encontramos los siguientes archivos:

1. **`agenda.vue`**:
    - Este archivo podría representar una página que muestre todos los eventos de una manera organizada, probablemente utilizando uno o más componentes de los que hemos visto, como el `Calendar.vue` o `Proximos.vue`. Esta página podría ser el lugar central donde los usuarios acceden a toda la información sobre los eventos.
2. **`event.vue`**:
    - Esta página probablemente se utiliza para mostrar los detalles de un solo evento, quizás al hacer clic en un evento desde el calendario o desde la lista de eventos. Es probable que se combine con el componente `Event.vue` para mostrar la información detallada del evento.
3. **`index.vue`**:
    - Este es probablemente el archivo principal o la página de inicio del proyecto. En muchos proyectos, la página `index.vue` actúa como la puerta de entrada, mostrando una visión general de los eventos o dirigiendo a los usuarios a otras secciones como el calendario o los eventos próximos.

Interacción entre Componentes y Páginas

1. **`Calendar.vue` y `MenuMeses.vue`**:
    - Estos componentes probablemente se usan juntos para crear un calendario interactivo. El componente `MenuMeses.vue` puede permitir a los usuarios navegar por diferentes meses, mientras que `Calendar.vue` muestra el calendario de eventos. Los usuarios pueden ver los eventos de un mes específico haciendo clic en un mes o navegando a través de las fechas.
2. **`Event.vue` y `agenda.vue`**:
    - El componente `Event.vue` se utiliza en la página `event.vue` para mostrar los detalles de un evento seleccionado. Por otro lado, `agenda.vue` puede usar componentes como `Proximos.vue` o `Calendar.vue` para mostrar los eventos en una vista más general o completa.
3. **`Proximos.vue` y `agenda.vue`**:
    - `Proximos.vue` probablemente se usa en la página `agenda.vue` para mostrar una lista de los próximos eventos de forma destacada. Este componente podría estar mostrando los eventos más cercanos en tiempo, lo cual es útil para los usuarios que desean ver rápidamente qué sucede en el futuro inmediato.

Resumen

- Los **componentes** son piezas reutilizables de la interfaz de usuario, como el calendario, la lista de eventos, y la vista detallada de los eventos.
- Las **páginas** son las vistas principales del proyecto, donde los componentes se integran para ofrecer la experiencia completa al usuario (por ejemplo, la página

principal `index.vue`, la página de agenda `agenda.vue`, y la página de detalles de un evento `event.vue`).

- Los componentes trabajan juntos, haciendo que la aplicación sea modular y escalable, permitiendo que diferentes partes del proyecto se gestionen de manera aislada pero de forma coherente.

Si deseas más detalles sobre cómo se usan específicamente estos componentes o si necesitas un análisis más profundo del código en algún archivo en particular, estaré encantado de ayudarte.

CALENDARIO

```vue
<template>
  <div>
    <v-sheet class="d-flex" height="54" tile>
      <v-select
        v-model="type"
        :items="types"
        class="ma-2"
        label="View Mode"
        variant="outlined"
        dense
        hide-details
      ></v-select>
      <v-select
        v-model="weekday"
        :items="weekdays"
        class="ma-2"
        label="Weekdays"
        variant="outlined"
        dense
        hide-details
      ></v-select>
    </v-sheet>
    <v-sheet>
      <v-calendar
        ref="calendar"
        v-model="value"
        :events="eventsFromStore"
        :view-mode="type"
        :weekdays="weekday"
      ></v-calendar>
    </v-sheet>
  </div>
</template>
```

Event

```vue
<template>
    <div>
        <h1>Productos</h1>
        <div v-if="getEvent.length">
            <ul >
                <li v-for="event in getEvents" :key="event.id">
                <h2>{{ event.title }}</h2>
                <p>{{ event.description }}</p>
                <strong>{{ event.price }} USD</strong>
                </li>
            </ul>
        </div>
    </div>
</template>
<script setup>
    // Instanciar el store
  const eventStore = useEventStore()

  // Obtener los productos
  const { getEvents } = eventStore;

</script>
```

Menú Meses

```vue
<template>
  <div class="text-center">
    <!-- Selección de Meses -->
    <v-select
      v-model="selectedMonth"  <!-- Aquí se almacena el mes seleccionado -->
      :items="months"          <!-- Los meses disponibles -->
      label="Seleccionar Mes"
    ></v-select>

    <!-- Menú de ubicación (se mantiene si lo necesitas) -->
    <v-menu :location="location">
      <template v-slot:activator="{ props }">
        <v-btn
          color="primary"
          dark
          v-bind="props"
        >
          Dropdown
        </v-btn>
      </template>

      <v-list>
        <v-list-item
          v-for="(item, index) in items"
          :key="index"
        >
          <v-list-item-title>{{ item.title }}</v-list-item-title>
        </v-list-item>
      </v-list>
    </v-menu>
  </div>
</template>
```

```
<script>
export default {
  data: () => ({
    // Meses disponibles
    months: [
      'Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',
      'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'
    ],
    selectedMonth: null,  // Aquí se almacena el mes seleccionado
    items: [
      { title: 'Click Me' },
      { title: 'Click Me' },
      { title: 'Click Me' },
      { title: 'Click Me 2' },
    ],
    locations: [
      'top',
      'bottom',
      'start',
      'end',
      'center',
    ],
    location: 'end',
  }),
}
</script>
```

Meses

```vue
<template>
  <v-card>
    <v-tabs
      bg-color="deep-purple-darken-4"
      center-active
    >
      <v-tab>Enero</v-tab>
      <v-tab>Febrero</v-tab>
      <v-tab>Marzo</v-tab>
      <v-tab>Abril</v-tab>
      <v-tab>Mayo</v-tab>
      <v-tab>Junio</v-tab>
      <v-tab>Julio</v-tab>
      <v-tab>Agosto</v-tab>
      <v-tab>Septiembre</v-tab>
      <v-tab>Octubre</v-tab>
      <v-tab>Noviembre</v-tab>
      <v-tab>Diciembre</v-tab>
    </v-tabs>
  </v-card>
</template>
```

```vue
<script>
import { useEventStore } from "@/stores/products.ts"; // Asegúrate de tener la ruta correcta

export default {
 data() {
   return {
     events: [], // Inicialmente vacío, se llenará con los datos del store
   };
 },
 mounted() {
   const eventStore = useEventStore();
   eventStore.fetchEvents().then(() => {
     console.log("Eventos desde el store:", eventStore.getevents); // Verifica la estructura de los eventos
     this.events = eventStore.getevents;
   });
 },
};

</script>
```

```
                <div class="px-4">
                  <v-switch
                    :label="`${isExpanded(event) ? 'Hide' : 'Show'} details`"
                    :model-value="isExpanded(event)"
                    density="compact"
                    inset
                    @click="() => toggleExpand(event)"
                  ></v-switch>
                </div>

                <v-divider></v-divider>

                <v-expand-transition>
                  <div v-if="isExpanded(event)">
                    <v-list :lines="false" density="compact">
                      <v-list-item :title="`📅 Event Date: ${event.date}`" active></v-list-item>
                    </v-list>
                  </div>
                </v-expand-transition>
              </v-card>
            </v-col>
          </v-row>
        </template>
      </v-data-iterator>
    </template>
```

Eventos Próximos

```vue
<template>
  <div v-if="events.length === 0">No hay eventos disponibles</div>
 <v-data-iterator
   :items="events"
   item-value="id"
 >
   <template v-slot:default="{ items, isExpanded, toggleExpand }">
     <v-row>
       <v-col
         v-for="event in items"
         :key="event.id"
         cols="12"
         md="6"
         sm="12"
       >
         <v-card>
           <v-card-title class="d-flex align-center">
             <v-icon
               color="#4CAF50"
               icon="mdi-calendar"
               size="18"
               start
             ></v-icon>

             <h4>{{ event.title }}</h4>
           </v-card-title>

           <v-card-subtitle>{{ event.date }}</v-card-subtitle>

           <v-card-text>
             {{ event.description }}
           </v-card-text>
```

Agenda

```vue
<template>
    <div>
        <h1>Agenda</h1>
        <Meses />
        <Calendar />
        <Proximos />
        <MenuMeses />
    </div>
</template>

<script setup>
import Meses from '~/components/Meses.vue';
import Calendar from '~/components/Calendar.vue';
import Proximos from '~/components/Proximos.vue';
import MenuMeses from '~/components/MenuMeses.vue';
</script>
```

```vue
<template>
    <div>
    <h1>Principal</h1>
    <Event  />
    </div>
</template>
<script setup>
  // Instanciar el store
  const eventStore = useEventStore()


  await useAsyncData('events', async () => {
    await eventStore.fetchEvents()
  })

</script>
```

Index

```vue
<template>
  <section class="flex flex-col md:flex-row justify-between items-center px-4 py-10">
    <div class="text-center md:text-left md:w-1/2">
      <h1 class="text-4xl font-bold text-blue-600">Bienvenido a EduPlanner</h1>
      <p class="mt-4 text-lg text-gray-700">Planifica tu educación de forma sencilla y eficiente.</p>
    </div>

    <div class="mt-6 md:mt-0 md:w-1/2 flex justify-center">

    </div>
  </section>
</template>

<style scoped>
section {
  background-color: #f3f4f6;
}

h1 {
  font-size: 2.5rem;
  color: #1c3d5a;
}

p {
  font-size: 1.2rem;
  color: #374151;
}
</style>
```

index.get,ts, de las carpetas server, api y events

```ts
export default defineEventHandler(async (event) => {
    const events = [
        {
            id: 1,
            title: 'Conferencia de Tecnología',
            date: '2023-11-25',
            description: 'Una conferencia sobre las últimas innovaciones en tecnología y desarrollo de software.',
        },
        {
            id: 2,
            title: 'Taller de Fotografía',
            date: '2023-12-02',
            description: 'Un taller práctico para aprender las bases de la fotografía digital y técnicas avanzadas.',
        },
        {
            id: 3,
            title: 'Concierto de Jazz en Vivo',
            date: '2023-12-10',
            description: 'Disfruta de una noche de jazz en vivo con artistas locales e internacionales.',
        },
        {
            id: 4,
            title: 'Exposición de Arte Moderno',
            date: '2023-12-15',
            description: 'Una muestra de obras de artistas contemporáneos de todo el mundo.',
        },
        {
            id: 5,
            title: 'Clase de Cocina Italiana',
            date: '2023-12-20',
            description: 'Aprende a cocinar auténticos platos italianos de la mano de un chef experimentado.',
        }

    ];

        return events;
    });
```