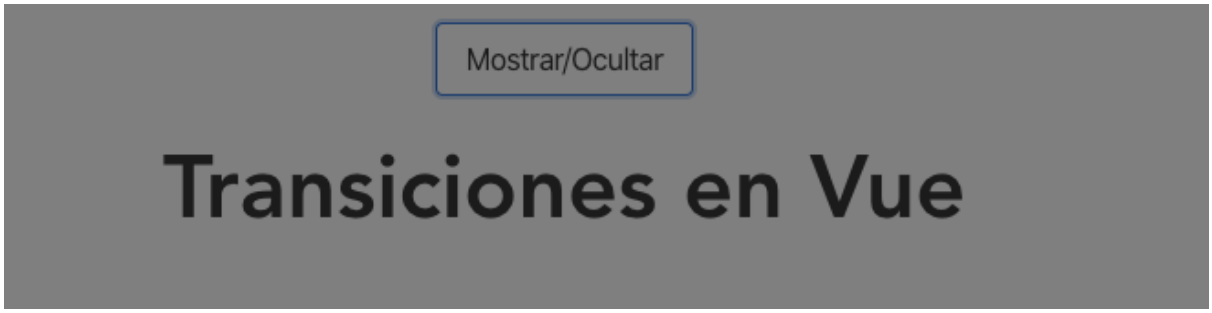
Transiciones

Nos permite realizar efectos cuando nos movamos entre elementos (que se esconda, que se oculte, etc..).

Partiremos del siguiente código en el app.vue:

```
<template>
  <div id="app">
    <button class="button" @click="show = !show">Mostrar/Ocultar</button>
    <br>
    <br>
    <h1 v-if="show" class="title is-1">Transiciones en Vue</h1>
  </div>
</template>


<script>
export default {
  name:'app',
  data (){
      return {
        show: 'true',
      }
  }
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font.smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  padding: 1em;
  width: 80%;
  margin: 0 auto;
}
</style>
```

Un botón que muestre u oculte un título.



En principio no se pueden agrupar varios elementos aunque más adelante veremos como, aquellos elementos que queramos que hagan una transición irán envueltos por la etiqueta <transition>.

E incluiremos

```
.fade-enter {
  opacity: 0;
}

.fade-enter-active {
  transition: all 1s cubic-bezier(1.0, 0.5, 0.8, 1.0);
}

.fade-enter-to {
  opacity: 1;
}

.v-leave {

}

.v-leave-active {

}

.v-leave-to {

}
```

En los estilos, además <transition name="fade">, el nombre fade substituye a la **v**.

Ahora para hacer animaciones funcionan así:

```
    <transition name="bounce">
    <h1 v-if="show" class="title is-1">Animaciones en Vue</h1>
    </transition>
```

y en los estilos:

```
.bounce-enter-active{
  animation: bounce-in .5s
}

.bounce-leave-active{
  animation: bounce-in .5s reverse
}

@keyframes bounce-in {
    0%{
      transform: scale(0);
    }
```

```
  50%{
    transform: scale(1.5);
  }
  100%{
    transform: scale(1);
  }
}
```

Animation, define la animación en tantos por cien y luego se la asignamos a los eventos de entrada y de salida.

El efecto que quiero añadir ahora es cuando yo entre en la página o la recargue que se añada la animación a ciertos componentes de la página, para ello añadiremos la propiedad **appear**:

```
<transition name="bounce" appear>
```

Ahora vamos a añadir el efecto de transición a elementos en grupo, para ello vamos a hacer una lista de elementos la cual cuando borremos algún elemento habrá una transición que afectará al grupo entero. Este es básicamente un ejemplo de la documentación oficial.
Definimos una lista de números hecha con un v-for.
Teniendo el app.vue:

```
<template>
  <div id="app">
    <button class="button" @click="add">Añadir</button>
    <button class="button" @click="remove">Eliminar</button>
    <ul class="list-group">
      <li v-for="number in numbers" :key="number" class="list-group-item">
        {{number}}
      </li>
    </ul>

    <br>
    <br>


  </div>
</template>

<script>
export default {
  name:'app',
  data (){
      return {
        show: 'true',
        numbers:[1,2,3,4,5,6,7,8,9],
        nextNum: 10,
      }
  },
  methods:{
```

```
    randomIndex: function(){
      return Math.floor(Math.random() * this.numbers.length)
    },
    add: function(){
      this.numbers.splice(this.randomIndex(),0,this.nextNum++)
    },
    remove: function(){
      this.numbers.splice(this.randomIndex(),1)
    },
  }
}
</script>
```

Ahora lo que vamos a hacer es añadir las transiciones para que no sea un cambio tan brusco el eliminar o añadir un elemento.

Para ello añadiremos un transition group:

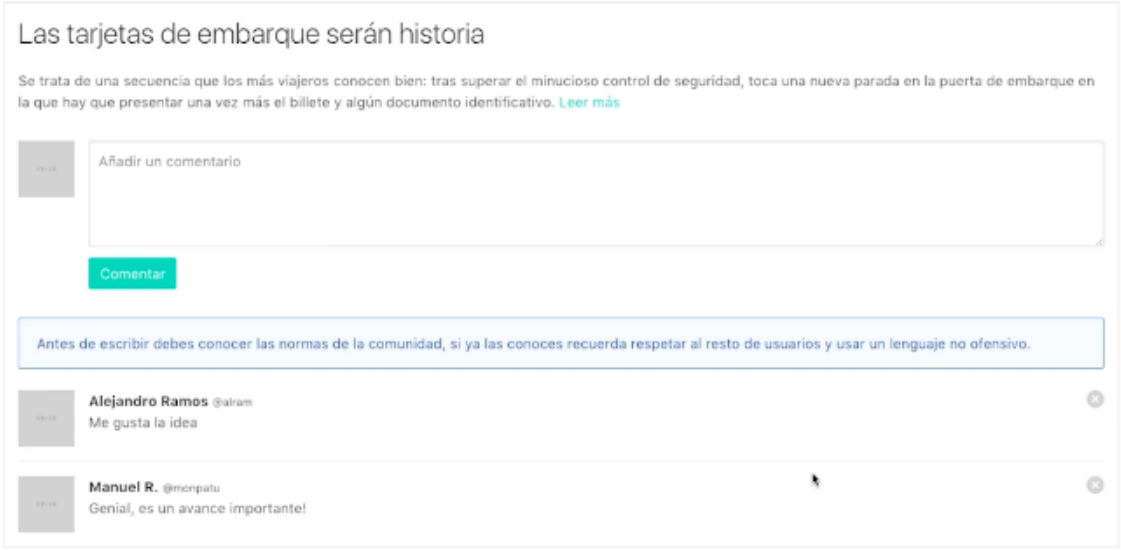
```
    <transition-group name="scale-fade" tag="ul" class="list-group" type="animation" >
      <li v-for="number in numbers" :key="number" class="list-group-item">
        {{number}}
      </li>
    </ul>
   </transition-group>
```

Y en las animaciones :

```
.scale-fade-move {
  transition: transform 1s;
}

.scale-fade-enter-active{
  animation: bounce-in .5s;
}

.scale-fade-leave-active {
  animation:bounce-in .5s reverse;
  position: absolute;
}

@keyframes bounce-in {
  0%{
    transform: scale(0);
  }
  50%{
    transform: scale(1.5);
  }
  100%{
    transform: scale(1);
  }
}
```

Ejercicio COMENTARIOS

Nuestro objetivo va a ser hacer algo parecido a lo siguiente:



Vamos a la faena, empezaremos con la siguiente base:


```
<template>
  <div id="app">
    <article>
      <h1 class="title">Las tarjetas de embarque serán historia </h1>
      <p>Se trata de una secuencia que los viajeros conocen bien: tras superar el minucioso control de seguridad, toca una nueva parada
        en la puerta de embarque en la que hay que presentar una vez mas el billete y algún documento identificativo</p>
    </article>

  </div>
</template>

<script>
export default {
  name:'app',

}
</script>

<style>
#app {
  color: #2c3e50;
  border: 1px solid #ddd;
```

```
  padding: 1em;
  width: 80%;
  margin: 5px auto;
}

</style>
```

Necesitaremos tres archivos más: CommentBox.vue, CommentForm.vue y SingleComment.vue.

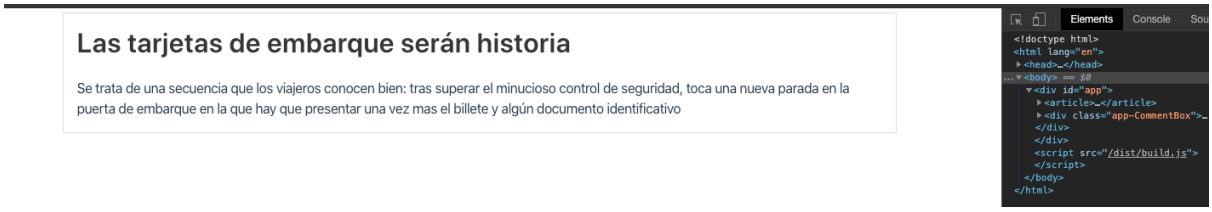
CommentBox.vue

```
<template>
  <div class="app-CommentBox">
      <div class="app-CommentBox_List">
      </div>
  </div>
</template>

<script>

export default {
  name: 'app-comment-box',
}
</script>
<style>

</style>
```

En el app.vue:

```
    <app-comment-box></app-comment-box>
  </div>
</template>

<script>
import AppCommentBox from './CommentBox.vue';

export default {
  name:'app',
  components: {
    AppCommentBox,
  }

}
```

Cuando lo añadamos aparecerá así, vemos en el inspector que ya aparece el componente:



Pasamos a importar este componente en el app.vue. En el commentbox tenemos que añadir el textbox y la zona de comentarios.

Vamos a generar datos falsos en el commentbox que luego retocaremos:

```
export default {
  name: 'app-comment-box',
  data() {
    return {
      comments : [
        {user:'Alejandro Rivas', userReference:'@alri',message:'Me gusta la idea'},
        {user:'Manuel Pérez', userReference:'@mape',message:'Lo apoyo, es una estupenda idea'},
      ]
    }
  }
}
```

Ya tenemos los comentarios en nuestro commentbox.

Ahora tenemos que cargar el singlecomment con los comentarios que vienen del commentbox, para ello habrá que definir los props necesarios.

```
<template>
  <article class="media">
    <figure class="media-left">
      <p class="image is-64x64">
        <img src="http://bulma.io/images/placeholders/128x128.png">
      </p>
    </figure>
    <div class="media-content">
      <div class="content">
        <p>
        </p>
      </div>
    </div>
    <div class="media-right">
    </div>
  </article>
</template>

<script>

export default {
  name: 'app-single-comment',
  data() {

  }
}
</script>
```

```
<style>

</style>
```

con props:

```
  name: 'app-single-comment',
    props: {
      comment:{
        type: Object,
        default: function(){
          return {message: 'texto por defecto'}
        },
        required: true,
      },
      index: {
        type: Number,
        default:100,
        required: true,
      }
    }
  }
```

Y dentro del párrafo:

```
       <p>
          <strong>{{ comment.user }}</strong><small>{{ comment.userReference }}</small>
          <br>
          {{ comment.message }}
        </p>
```

para pasarlos al html.

Y en el commentbox.vue

```
<div class="app-CommentBox_List">
        <app-single-comment
          v-for="(comment,index) in comments"
          :comment="comment"
          :index="index"
          :key="index"
          >
        </app-single-comment>
      </div>
```

realizando el consiguiente import:

```
import AppSingleComment from './SingleComment.vue';
```

Y en el commentbox añado los componentes:

```
export default {
  name: 'app-comment-box',
  data() {
    return {
```

```
    comments : [
      {user:'Alejandro Rivas', userReference:'@alri',message:'Me gusta la idea'},
      {user:'Manuel Pérez', userReference:'@mape',message:'Lo apoyo, es una estupenda idea'},
    ]
  }
 },
 components:{
  AppSingleComment
 }
}
```

Quedando el siguiente aspecto.

Ahora que ya tenemos el esqueleto vamos a empezar a añadir botones de eliminar y transiciones. Para borrar un comentario vamos a hacer uso de los eventos.

El botón:

```
<div class="media-right">
   <button
   class="delete"
   @click="EraseComment"
   >
  </button>
  </div>
```

Y el método, que emitirá el evento pasando el índice y llamando al método remove en comment box.

```
methods:{
  EraseComment(index) {
    this.$emit('Remove',this.index);
  }
}
```

En el commentbox crearemos el método Erase.

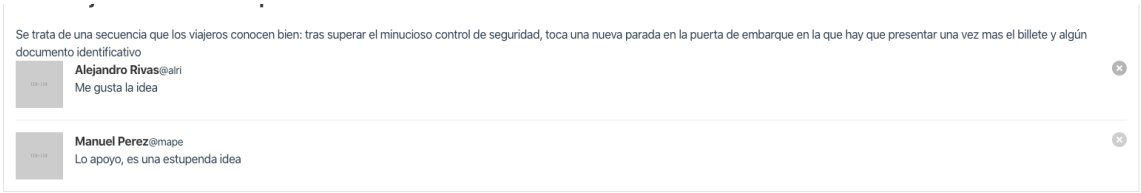
```
methods: {
  removeComment(index){
```

```
      this.comments.splice(index,1);
    }
  }
```

Y más arriba, hacemos la llamada a removeComment.

```
key="index"
          @remove="removeComment"
          >
```

Quedando así:



Ahora vamos añadir el textarea para añadir comentarios. Para eso el commentForm tendrá esta forma:

```
<template>
  <article class="media">
    <figure class="media-left">
      <p class="image is-64x64">
        <img src="http://bulma.io/images/placeholders/128x128.png">
      </p>
    </figure>
    <div class="media-content">
      <div class="field">
        <p class="control">
          <textarea
          class="textarea"
          placeholder="Escribe tu mensaje"
          v-model="message"
          >
        </textarea>
        </p>
      </div>
    </div>
    <nav class="level">
      <div class="level-left">
        <div class="level-item">
        </div>
      </div>
    </nav>
  </article>
</template>

<script>

export default {
  data() {
    return {
```

```
    name: 'Random',
    userReference: '@reference',
    message:'',
   }
  }

}

</script>
<style scoped>
.media {
  padding: 1em 0;
  margin: 1em 0;
}

</style>
```

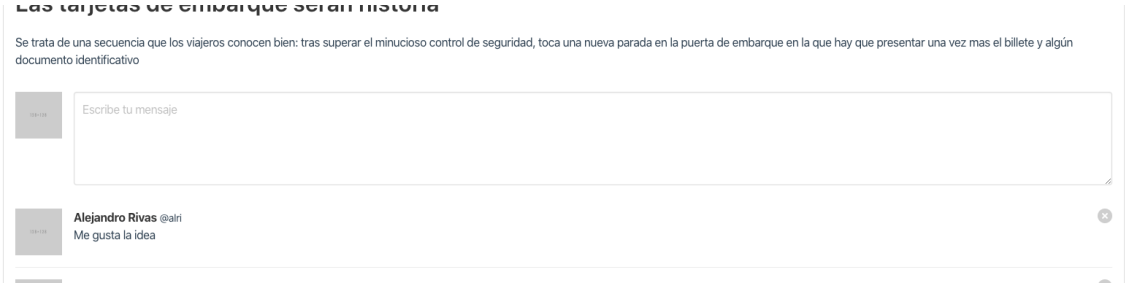
En el commentBox añadiré otro import:

```
import AppCommentForm from './CommentForm.vue';
```

lo Añado también:

```
  components:{
    AppSingleComment,
    AppCommentForm
  },
```

Se vería mas o menos así:



Ahora voy a incluir el botón de añadir en el commentForm:

```
<div class="level-item">
          <a class="button is-primary" @click="addComment">Comentar</a>
        </div>
```

Y como método para enviar la información

```
  methods: {
    addComment() {
      this.$emit('newComment',this.name, this.userReference, this.message);
      this.message= '';
    }
  }
```

En el commentbox:

```
methods: {
    removeComment(index) {
      this.comments.splice(index, 1);
    },
    addComment(user, userReference, message){
      this.comments.push ({
        user,
        userReference,
        message,
      });
    }
  }
}
```

Y en el componente recogemos el evento:

```
  <app-comment-form @newComment="addComment"></app-comment-form>
```

Con esto añadimos comentarios.

Por último vamos a añadir las transiciones.

Crearemos un transition-group:

```
<transition-group name="fade">
                <app-single-comment
                  v-for="(comment, index) in comments"
                  :comment="comment"
                  :index="index"
                  :key="index"
                  @Remove="removeComment"
                >
                </app-single-comment>
        </transition-group>
```

Y los efectos:

```
.fade-enter-active,
.fade-leave-active {
  transition: opacity .5s;
}
.fade-enter,
.fade-leave-to {
  opacity: 0;
}
```

Aunque no hemos metido animaciones.