https://vee-validate.logaretm.com/v4/guide/components/handling-forms/

https://www.npmjs.com/package/yup

Загальна форма

```
<template>
  <Form
     @submit="onSubmit"
     :validation-schema="schema"
     :initial-values="formValues"
   >
      <div>
        <label for="поле">Title:</label>
        <Field name="поле " type="тип-поля" />
        <ErrorMessage name="поле " />
      </div>
      <-- ... інші поля ... -->
  </Form>
</template>
<script>
//---- імпортк компонентів -----------
import { Form, Field, ErrorMessage }
                                from 'vee-validate'
import * as yup from 'yup';

//---- опис схеми (правил валідації для полів)
const schema = yup.object({
  поле: yup.правило1().правило2 (),
  // ...
});

function onSubmit(values, { resetForm }) {
  console.log(values); // send data to API
  // reset the form and the field values to their
initial values
  resetForm();
```

Приклад

```
<template>
  <products-layout>
    <Form
      v-slot="{ handleReset }"
      :validation-schema="schema"
      :initial-values="product"
      @submit="onSubmit"
    >
      <div>
        <label for="title">Title:</label>
        <Field name="title" type="text" />
        <ErrorMessage name="title" />
      </div>
      <div>
        <label for="price">Price:</label>
        <Field name="price" type="number" />
        <ErrorMessage name="price" />
      </div>
      <button type="submit">Submit</button>
      <button type="button"
@click="handleReset">Reset</button>
    </Form>
  </products-layout>
</template>

<script>
import { Form, Field, ErrorMessage } from 'vee-validate'
import * as yup from 'yup'
import { mapActions, mapGetters } from 'vuex'

const initialProductData = {
```

```
}
</script>
```

```
  title: '',
  price: null,
}

export default {
  components: {
    Form,
    Field,
    ErrorMessage,
  },

  props: {
    id: {
      type: String,
      required: false,
    },
  },
  data() {
    return {
      product: { ...initialProductData },
      schema: yup.object({
        title: yup.string().required(),
        price: yup.number().required().min(1),
      }),
    }
  },
  computed: {
    ...mapGetters('products', ['getCurrentItem',
'isLoading', 'hasError']),
  },

  watch: {
    getCurrentItem: {
      handler(newValue) {
        if (newValue) {
          this.product = {
            title: newValue.title,
            price: newValue.price,
```

```
        }
        this.updateKey++
      }
    },
    deep: true,
    immediate: true,
  },
},

methods: {
  ...mapActions('products', ['loadItemDataById',
'addItem', 'updateItem']),
  async onSubmit(values, { resetForm }) {
    try {
      if (this.id) {
        await this.updateItem({ itemId: this.id, data:
values })
        console.log('Product updated:', values)
      } else {
        await this.addItem(values)
        console.log('Product created:', values)
      }

      resetForm()
      this.$router.push({
        name: 'product-list',
      })
    } catch {
      alert()
    }
  },
},
created() {
  this.product = { ...initialProductData }

  if (this.id) {
    this.loadItemDataById(this.id)
  }
```

	`` }, } </script> ```