# LAB: Admin Features

NEXT: LAB: SPINNER, CART & STRORE SUBSCRIPTIONS

## Products Table

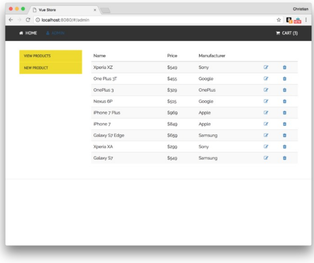
Remember in the admin section, we already added a form for creating new products. As an admin, I should be able to view the products in a table. Let's see about that:

```
// src/pages/Admin/Products
<template>
    <div class="col-lg-9 col-md-9 col-sm-12 col-xs-12">
          <table class="table table-striped">
            <thead>
            <tr>
              <th>Name</th>
              <th>Price</th>
              <th>Manufacturer</th>
              <th></th>
              <th></th>
            </tr>
            </thead>
            <tbody>
            <tr v-for="product in products">
              <td>{{product.name}}</td>
              <td>${{product.price}}</td>
              <td>{{product.manufacturer.name}}</td>
              <td><router-link :to="'/admin/edit/'+product._id"><i class="fa fa-pencil-square-o"></i></router-link></td>
              <td><a @click="deleteProduct(product._id)" ><i class="fa fa-trash"></i></a></td>
            </tr>
            </tbody>
          </table>
        </div>
</template>

<script>
  export default {
    computed: {
      products () {
        return this.$store.getters.allProducts
      }
    },
    created () {
      if (this.products.length === 0) {
        this.$store.dispatch('allProducts')
      }
    },
    methods: {
      deleteProduct (id) {
        this.$store.dispatch('removeProduct', id)
      }
    }
  }
</script>
```

We created a index page for the admin which points to `Products` component. That's the component right above. It displays all the products in a table with a link to edit/update a product and a button to remove a product on each row.

The button to remove a product calls `deleteProduct` which in turn dispatches the `removeProduct` action.




## Editing Products

The edit link from the `Products` component takes you to an edit page which should have a form pre-loaded with the selected component. There, you can update the selected component. This is what the Edit page's component looks like:

```
<template>
  <product-form
    @save-product="updateProduct"
    :model="model"
    :manufacturers="manufacturers"
    :isEditing="true" ></product-form>
</template>

<script>
  import ProductFrom from '../../components/product/ProductForm.vue'
  export default {
    created () {
      if (!this.model.name) {
        console.log('dispatched')
        this.$store.dispatch('productById', this.$route.params['id'])
      }
      if (this.manufacturers.length === 0) {
        this.$store.dispatch('allManufacturers')
      }
    },
    computed: {
      manufacturers () {
        return this.$store.getters.allManufacturers
      },
      model () {
        const productById = this.$store.getters.productById(this.$route.params['id'])
        return Object.assign({}, productById)
      }
    },
    methods: {
      updateProduct (model) {
        console.log('model', model)
        this.$store.dispatch('updateProduct', model)
      }
    },
    components: {
      'product-form': ProductFrom
    }
  }
</script>
```

Another case of re-use. We are re-using the `ProductForm` component which was first used in the `New` page component. This time, it's initialized with a `model` (existing product) and the `isEditing` flag is raised as well.

When the button on the form is clicked, `updateProduct` action is dispatched with the model as payload. This payload is sent the server.