```
import { Button, Form, notification, Popconfirm, Spin } from "antd";
import React, { useEffect, useMemo, useState } from "react";

import { useDispatch, useSelector } from "react-redux";
import Utils from "utils";
import MediTable from "components/Custom";
import EditableCell from "components/EditableCell";
import {
  EditOutlined,
  DeleteOutlined,
  PlusOutlined,
  SaveOutlined,
  CloseOutlined,
} from "@ant-design/icons";
import {
  DeleteDonHangKhamSucKhoeKhac,
  GetAllDonHangKhamSucKhoeVatTu,
  UpsertDonHangKhamSucKhoeKhac,
} from "store/slices/khamSucKhoeDoanSlice";
import { isEmpty } from "lodash";

export const initHoSo = {
  action: "initial",
  isRequired: true,
};

const VatTu = ({ donHangId }) => {
  const { loading, DonHangKhamSucKhoeVatTuList } = useSelector(
    (state) => state.ksk
  );
  const dispatch = useDispatch();
  const [form] = Form.useForm();
  useEffect(() => {
    if (isEmpty(donHangId)) return;

    dispatch(
      GetAllDonHangKhamSucKhoeVatTu({
        id: donHangId,
        loai: 2,
      })
    );
  }, [dispatch, donHangId]);
  const [editingKey, setEditingKey] = useState("");

  // eslint-disable-next-line react-hooks/exhaustive-deps
  const edit = (record) => {
    form.setFieldsValue({
      tenVatTu: null,
      ...record,
    });
    setEditingKey(record.action === "initial" ? record.action : record.id);
  };

  // eslint-disable-next-line react-hooks/exhaustive-deps
  const save = async (key) => {
    try {
      const row = await form.validateFields();
      const newData = DonHangKhamSucKhoeVatTuList.slice(0);
      const index = newData.findIndex((item) =>
        item.id ? item.id === key : item.action === key
      );
      const item = newData[index];
      const payload = {
        id: item?.id || null,
        tenVatTu: item?.tenVatTu,
        loai: 2,
        donHangId: donHangId,
        ...row,
        onSuccess: ({ data }) => {
          if (data.isOk) {
            dispatch(
```

```
            GetAllDonHangKhamSucKhoeVatTu({
              id: donHangId,
              loai: 2,
            })
          );
          setEditingKey("");
        } else {
          notification.error({
            message: "Lưu chi tiết",
            description: data.description,
          });
        }
      },
    };

    dispatch(UpsertDonHangKhamSucKhoeKhac(payload));
  } catch (error) {}
};

// eslint-disable-next-line react-hooks/exhaustive-deps
const isEditing = (record) =>
  record.action === "initial"
    ? record.action === editingKey
    : record.id === editingKey;

const tableColumns = useMemo(
  () => [
    {
      title: "STT",
      align: "center",
      width: "80px",
      render: (_, __, index) => index + 1,
    },
    {
      title: "Tên vật tư",
      dataIndex: "tenVatTu",
      editable: true,
    },
    {
      title: "Ngày tạo",
      dataIndex: "createdAt",
      width: "140px",
      render: (value) => value && Utils.formatDate(value),
    },
    {
      fixed: "right",
      align: "center",
      width: "120px",
      render: (_, record, index) => {
        const editable = isEditing(record);
        return editable ? (
          <>
            <Button
              onClick={() => save(record?.id || "initial")}
              className="mr-2"
              icon={<SaveOutlined />}
              shape="circle"
            />
            <Button
              onClick={() => setEditingKey("")}
              className="mr-2"
              icon={<CloseOutlined />}
              shape="circle"
            />
          </>
        ) : (
          <>
            <Button
              title={
                record.action === "initial"
                  ? "Thêm thông tin"
                  : "Sửa thông tin"
              }
              onClick={() => edit(record)}
```

```
              className="mr-2"
              icon={
                record.action === "initial" ? (
                  <PlusOutlined />
                ) : (
                  <EditOutlined />
                )
              }
              shape="circle"
            />
            {record.id && (
              <>
                <Popconfirm
                  title="Bạn có muốn xóa không?"
                  placement="topLeft"
                  onConfirm={() => {
                    dispatch(
                      DeleteDonHangKhamSucKhoeKhac({
                        id: record.id,
                        onSuccess: ({ data }) => {
                          if (data.isOk) {
                            dispatch(
                              GetAllDonHangKhamSucKhoeVatTu({
                                id: donHangId,
                                loai: 2,
                              })
                            );
                            setEditingKey("");
                          } else {
                            notification.error({
                              message: "Lưu chi tiết",
                              description: data.description,
                            });
                          }
                        },
                      })
                    );
                  }}
                >
                  <Button
                    title={"Xóa thông tin"}
                    className="mr-2"
                    icon={<DeleteOutlined type="primary" />}
                    shape="circle"
                  />
                </Popconfirm>
              </>
            )}
          </>
        );
      },
    },
  ],
  [dispatch, donHangId, edit, isEditing, save]
);

const mergedColumns = tableColumns.map((col) => {
  if (!col.editable) {
    return col;
  }

  return {
    ...col,
    onCell: (record) => ({
      record,
      title: col.title,
      dataIndex: col.dataIndex,
      editing: isEditing(record),
      inputType: col?.inputType,
      isRequired: col?.isRequired,
      options: col?.options,
    }),
  };
});
```

```
  return (
    <Spin tip="Đang tải..." spinning={loading}>
      <Form form={form} component={false}>
        <MediTable
          components={{
            body: {
              cell: EditableCell,
            },
          }}
          tableColumns={mergedColumns}
          dataSource={[{ action: "initial" }].concat(
            DonHangKhamSucKhoeVatTuList.map((x) => ({
              ...x,
            }))
          )}
          scroll={{ x: "max-content" }}
          rowKey={(item) => item?.id}
        />
      </Form>
    </Spin>
  );
};

export default VatTu;
```