```
import { Button, Form, notification, Popconfirm, Spin } from "antd";
import React, { useEffect, useMemo, useState } from "react";
import {
  EditOutlined,
  DeleteOutlined,
  PlusOutlined,
  SaveOutlined,
  CloseOutlined,
} from "@ant-design/icons";
import { useDispatch, useSelector } from "react-redux";
import {
  DeletePhuongAnKinhDoanhChiTiet,
  GetPhuongAnKinhDoanhChiTiet,
  UpsertPhuongAnKinhDoanhChiTiet,
} from "store/slices/khamSucKhoeDoanSlice";
import Utils from "utils";
import { PermitValue, permitKey } from "constants";
import MediTable from "components/Custom";
import EditableCell from "components/EditableCell";
import { guidEmpty } from "constants";

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

const ChiTietPhuongAn = ({ phuongAnId, PhuongAnKinhDoanhDetail }) => {
  const dispatch = useDispatch();
  const { loading, PhuongAnKinhDoanhChiTietList, DmDichVuList } = useSelector(
    (state) => state.ksk
  );

  const { permitList } = useSelector((state) => state.auth);
  const [form] = Form.useForm();
  const [editingKey, setEditingKey] = useState("");

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

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

  // eslint-disable-next-line react-hooks/exhaustive-deps
  const save = async (key) => {
    try {
      if (
        PhuongAnKinhDoanhDetail?.totalSigns !== null &&
        PhuongAnKinhDoanhDetail?.totalSigned !== null &&
        PhuongAnKinhDoanhDetail?.totalSigned ===
          PhuongAnKinhDoanhDetail?.totalSigns
      ) {
        notification.error({
          message: "Lưu phương án",
          description: "Phương án đã được ký duyệt, bạn không thể sửa",
        });
        return;
      }

      const row = await form.validateFields();
      const newData = PhuongAnKinhDoanhChiTietList.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,
      phuongAnId: phuongAnId,
      soluong: PhuongAnKinhDoanhDetail?.soLuotKham,
      donGia: item?.donGia !== null ? item?.donGia : 0,
      donGiaThucTe: item?.donGiaThucTe !== null ? item?.donGiaThucTe : 0,
      ptChietKhau: PhuongAnKinhDoanhDetail?.ptChietKhau,
      ...row,
      onSuccess: () => {
        dispatch(
          GetPhuongAnKinhDoanhChiTiet(
            phuongAnId !== null && phuongAnId !== undefined
              ? phuongAnId
              : guidEmpty
          )
        );
        setEditingKey("");
      },
    };

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

// eslint-disable-next-line react-hooks/exhaustive-deps
const allowView = (value) =>
  Utils.checkPermitValue(value, permitList, permitKey.crm_khamsuckhoedoan);

useEffect(() => {
  dispatch(
    GetPhuongAnKinhDoanhChiTiet(
      phuongAnId !== null && phuongAnId !== undefined ? phuongAnId : guidEmpty
    )
  );

  // DmDichVuList;
}, [dispatch, phuongAnId]);

const Footer = () => {
  let tongCong = 0;
  for (
    let index = 0;
    index <= PhuongAnKinhDoanhChiTietList.length - 1;
    index++
  ) {
    tongCong += PhuongAnKinhDoanhChiTietList[index].thanhTien;
  }

  let tongCongThucTe = 0;
  let chietKhauThucTe = 0;
  for (
    let index = 0;
    index <= PhuongAnKinhDoanhChiTietList.length - 1;
    index++
  ) {
    tongCongThucTe += PhuongAnKinhDoanhChiTietList[index].thanhTienThucTe;
  }

  if (PhuongAnKinhDoanhDetail?.ptChietKhau > 0) {
    chietKhauThucTe =
      (tongCongThucTe * PhuongAnKinhDoanhDetail?.ptChietKhau) / 100;
  }

  return (
    <>
      <h5>Tổng cộng: ({Utils.formatterNumber(tongCong)})</h5>
      <h5>Tổng cộng thực tế: ({Utils.formatterNumber(tongCongThucTe)})</h5>
      <h5>
        Chênh lệch HĐ: ({Utils.formatterNumber(tongCong - tongCongThucTe)})
```

```
      </h5>
      <h5>
        Phí chênh lệch HĐ: (
        {PhuongAnKinhDoanhDetail?.ptTyLeChenh > 0
          ? Utils.formatterNumber(
              (PhuongAnKinhDoanhDetail?.ptTyLeChenh *
                (tongCong - tongCongThucTe)) /
                100
            )
          : "0"}
        )
      </h5>
      <h5>
        Chiết khấu
        {` (${PhuongAnKinhDoanhDetail?.ptChietKhau}%)`}: (
        {PhuongAnKinhDoanhDetail?.ptChietKhau > 0
          ? Utils.formatterNumber(chietKhauThucTe)
          : "0"}
        )
      </h5>
      <h5>
        Thu thực tế: (
        {Utils.formatterNumber(
          tongCongThucTe +
            (PhuongAnKinhDoanhDetail?.ptTyLeChenh *
              (tongCong - tongCongThucTe)) /
              100
        )}
        )
      </h5>
    </>
  );
};

const tableColumns = useMemo(
  () => [
    {
      title: "STT",
      align: "center",
      render: (_, __, index) => index + 1,
      width: "50px",
    },
    {
      title: "Dịch vụ",
      dataIndex: "dichVuId",
      editable: true,
      inputType: "select",
      width: "350px",
      options: DmDichVuList.map((i) => ({
        id: i.id,
        name: i.tenDichVu,
      })),
      render: (value, record) => record.tenDichVu,
    },
    {
      title: "Đơn giá HĐ",
      dataIndex: "donGia",
      editable: true,
      inputType: "number",
      width: "110px",
      render: (value, record) => Utils.formatterNumber(record.donGia),
    },
    {
      title: "Thành tiền HĐ",
      dataIndex: "thanhTien",
      editable: false,
      width: "110px",
      render: (value, record) => Utils.formatterNumber(record.thanhTien),
    },
    {
      title: "Giá thực tế",
      dataIndex: "donGiaThucTe",
      editable: true,
      inputType: "number",
```

```
  width: "110px",
  render: (value, record) => Utils.formatterNumber(record.donGiaThucTe),
},
{
  title: "Tiền thực tế",
  dataIndex: "thanhTienThucTe",
  editable: false,
  width: "110px",
  render: (value, record) =>
    Utils.formatterNumber(record.thanhTienThucTe),
},

{
  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"
          }
          disabled={
            (record.action === "initial"
              ? !allowView(PermitValue.them)
              : !allowView(PermitValue.sua)) ||
            PhuongAnKinhDoanhDetail?.totalSigned > 1
          }
          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"
              disabled={
                !allowView(PermitValue.xoa) ||
                PhuongAnKinhDoanhDetail?.totalSigned > 1
              }
              onConfirm={() => {
                dispatch(
                  DeletePhuongAnKinhDoanhChiTiet({
                    id: record.id,
                    onSuccess: () => {
                      dispatch(
                        GetPhuongAnKinhDoanhChiTiet(
```

```
                            phuongAnId !== null ? phuongAnId : guidEmpty
                          )
                        );
                      },
                    })
                  );
                }}
              >
                <Button
                  title={"Xóa thông tin"}
                  disabled={
                    !allowView(PermitValue.xoa) ||
                    PhuongAnKinhDoanhDetail?.totalSigned > 1
                  }
                  className="mr-2"
                  icon={<DeleteOutlined type="primary" />}
                  shape="circle"
                />
              </Popconfirm>
            </>
          )}
        </>
      );
    },
  },
],
[
  DmDichVuList,
  PhuongAnKinhDoanhDetail?.totalSigned,
  allowView,
  dispatch,
  edit,
  isEditing,
  phuongAnId,
  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(
          PhuongAnKinhDoanhChiTietList.map((x) => ({
            ...x,
          }))
        )}
        scroll={{ x: "max-content" }}
```

```
          rowKey={(item) => item?.id}
          onRenderFooter={() => Footer()}
        />
      </Form>
    </Spin>
  );
};

export default ChiTietPhuongAn;
```