```
import React, {
  useState,
  useEffect,
  useMemo,
  useRef,
  useCallback,
} from "react";
import { useDispatch, useSelector } from "react-redux";
import { Button, Popconfirm, Select, Spin, Tag, Tooltip } from "antd";
import "react-circular-progressbar/dist/styles.css";

import MeditechTablePage, {
  FilterComponent,
} from "components/table-layout/index";
import Utils from "utils";
import MediTable from "components/Custom";

import {
  EditOutlined,
  DeleteOutlined,
  ReadOutlined,
  EyeOutlined,
} from "@ant-design/icons";
import {
  deleteNoti,
  getSearchNoti,
  readNoti,
} from "store/slices/common/erpNotificationSlice";
import { ERP_NOTIFICATION_TYPE } from "constants";
import { DETAIL_PAGE_PATH } from "configs/AppConfig";
const { Option } = Select;

const initSearch = {
  fromTime: null,
  toTime: null,
  searchText: "",
  notificationType: null,
};

const NotiReport = () => {
  const dispatch = useDispatch();

  const searchFormRef = useRef(initSearch);
  const [searchFormData, setSearchFormData] = useState(searchFormRef.current);

  const {
    dataSearch: { data, loading },
  } = useSelector((state) => state.erpNotification);
  useEffect(() => {
    dispatch(getSearchNoti({ ...searchFormRef.current }));
  }, [dispatch]);

  useEffect(() => {}, []);

  const reloadData = useCallback(() => {
    dispatch(getSearchNoti({ ...searchFormRef.current }));
  }, [dispatch]);

  const searchHandle = (dataSearch) => {
    const newDataSearch = {
      ...dataSearch,
      startDate: dataSearch.startDate === "" ? null : dataSearch.startDate,
      endDate: dataSearch.endDate === "" ? null : dataSearch.endDate,
    };
    searchFormRef.current = { ...searchFormRef.current, ...newDataSearch };
    setSearchFormData({ ...searchFormRef.current });
    reloadData();
  };
  const handleDelete = useCallback(
    (record) => {
      const payload = {
```

```
      id: record?.receivedId,
      onSuccess: () => {
        reloadData();
      },
    };
    dispatch(deleteNoti(payload));
  },
  [reloadData, dispatch]
);
const handleMarkRead = (record) => {
  const payload = {
    id: record?.notiId,
    onSuccess: () => {
      reloadData();
    },
  };
  dispatch(readNoti(payload));
};
const handleViewDetail = (record) => {
  const url = `${DETAIL_PAGE_PATH}?page=${record.objectType}&${record.objectType}Id=${record.objectId}`;
  window.open(url, "_blank", "noopener,noreferrer");
};
const columns = useMemo(
  () => [
    {
      title: "STT",
      dataIndex: "index",
      align: "center",
      fixed: "left",
      render: (_, __, index) => index + 1,
    },

    {
      title: "Chủ đề",
      dataIndex: "subject",
      key: "subject",
    },
    {
      title: "Nội dung",
      dataIndex: "content",
      key: "content",
    },
    {
      title: "Loại",
      align: "center",
      dataIndex: "type",
      key: "type",

      render: (type) => {
        const option = ERP_NOTIFICATION_TYPE.find(
          (opt) => opt.value === type
        );
        return option ? option.name : "Chưa cập nhật";
      },
    },
    {
      title: "Người gửi",
      dataIndex: "senderName",
      key: "name",
    },
    {
      title: "Thời gian gửi",
      align: "center",
      dataIndex: "sendTime",
      render: (date) => date && Utils.formatDate(date),
    },
    {
      title: "Đã đọc",
      align: "center",
      dataIndex: "isRead",
      render: (item) => {
        return item ? (
          <Tag color="green">Đã đọc</Tag>
        ) : (
```

```
          <Tag color="red">Chưa đọc</Tag>
        );
      },
    },
    {
      name: "action",
      fixed: "right",
      align: "center",
      render: (_, record) => (
        <>
          <Button
            title={"Xem chi tiết"}
            className="mr-2"
            icon={<EyeOutlined />}
            shape="circle"
            onClick={() => {
              handleViewDetail(record);
            }}
          />
          <Button
            title={"Đánh giấu đã đọc"}
            className="mr-2"
            icon={<ReadOutlined />}
            shape="circle"
            onClick={() => {
              handleMarkRead(record);
            }}
            disabled={record.isRead}
          />

          <Popconfirm
            title="Bạn có chắc muốn xóa không?"
            placement="topLeft"
            disabled={false}
            onConfirm={() => handleDelete(record)}
          >
            <Button
              title={"Xóa thông báo"}
              className="mr-2"
              icon={<DeleteOutlined />}
              shape="circle"
              disabled={false}
            />
          </Popconfirm>
        </>
      ),
    },
  ],
  [handleDelete]
);

const onRenderSearchView = () => (
  <FilterComponent
    datasource={searchFormData}
    onSearchImmediate={searchHandle}
    renderInputSearch
    renderDatePicker
  >
    <Select
      allowClear
      key="notificationType"
      data-field="notificationType"
      placeholder="Loại"
    >
      {ERP_NOTIFICATION_TYPE.map((item) => (
        <Option key={item.value} value={item.value}>
          {item.name}
        </Option>
      ))}
    </Select>
  </FilterComponent>
);

return (
```

```
    <>
      <MeditechTablePage onRenderSearchView={onRenderSearchView}>
        <Spin tip="Đang tải..." spinning={loading}>
          <MediTable
            dataSource={data?.notificationList}
            tableColumns={columns}
            rowKeyField={(item) => item.id}
          />
        </Spin>
      </MeditechTablePage>
    </>
  );
};

export default NotiReport;
```