```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import TraHangService from "services/sale/TraHangService";

//import { initTraHangCt } from "views/app-views/pharma/managements/TraHang/BanLe";

export const getDSTraHang = createAsyncThunk(
  "TraHang/getDSTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const response = await TraHangService.searchGrid(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getTraHangByMaTraHang = createAsyncThunk(
  "TraHang/getTraHangByMaTraHang",
  async (id, { rejectWithValue }) => {
    try {
      const response = await TraHangService.getTraHangBymatrahang(id);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getPhieuBanHangByKhachHang = createAsyncThunk(
  "TraHang/getPhieuBanHangByKhachHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.getPhieuBanHangByKhachHang(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getPhieuTraByPhieuBan = createAsyncThunk(
  "TraHang/getPhieuTraByPhieuBan",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.getPhieuTraByPhieuBan(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const createTraHang = createAsyncThunk(
  "TraHang/createTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.create(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
```

```
  }
);

export const updateTraHang = createAsyncThunk(
  "TraHang/updateTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.update(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const changeStateTraHang = createAsyncThunk(
  "TraHang/changeStateTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.changeStateTraHang(payload);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const deleteTraHang = createAsyncThunk(
  "TraHang/deleteTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await TraHangService.deleteTraHang(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getTraHangCTByMaTraHang = createAsyncThunk(
  "TraHang/getTraHangCTByMaTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.getTraHangCTBymatrahang(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getTonTraHang = createAsyncThunk(
  "TraHang/getTonTraHang",
  async (data, { rejectWithValue }) => {
    try {
      const response = await TraHangService.getTonTraHang(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
```

```
);

export const addNewTraHangCT = createAsyncThunk(
  "TraHang/addNewTraHangCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.addTraHangCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const updateTraHangCT = createAsyncThunk(
  "TraHang/updateTraHangCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await TraHangService.updateTraHangCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const deleteTraHangCT = createAsyncThunk(
  "TraHang/deleteTraHangCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const response = await TraHangService.deleteTraHangCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  TraHangList: [],
  TraHangDetail: {},
  PhieuBanHangByKhachList: [],
  PhieuTraByPhieuBan: [],
};

export const TraHangSlice = createSlice({
  name: "trahang",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
    setTraHangCTList: (state, action) => {
      //state.TraHangCTList = action.payload;
      state.PhieuTraByPhieuBan = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getDSTraHang.pending, (state) => {
        state.loading = true;
      })
```

```
.addCase(getDSTraHang.fulfilled, (state, action) => {
  state.loading = false;
  state.TraHangList = action.payload;
})
.addCase(getDSTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(getPhieuBanHangByKhachHang.fulfilled, (state, action) => {
  state.loading = false;
  state.PhieuBanHangByKhachList = action.payload;
})
.addCase(getPhieuTraByPhieuBan.fulfilled, (state, action) => {
  state.loading = false;
  state.PhieuTraByPhieuBan = action.payload;
})
.addCase(getTraHangByMaTraHang.pending, (state) => {
  state.loading = true;
})
.addCase(getTraHangByMaTraHang.fulfilled, (state, action) => {
  state.loading = false;
  state.TraHangDetail = action.payload;
})
.addCase(getTraHangByMaTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(createTraHang.pending, (state) => {
  state.loading = true;
})
.addCase(createTraHang.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(createTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(updateTraHang.pending, (state) => {
  state.loading = true;
})
.addCase(updateTraHang.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(updateTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(deleteTraHang.pending, (state) => {
  state.loading = true;
})
.addCase(deleteTraHang.fulfilled, (state, action) => {
  state.loading = false;
})
.addCase(deleteTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(getTonTraHang.pending, (state) => {
  state.loading = true;
})
.addCase(getTonTraHang.fulfilled, (state, action) => {
  state.loading = false;
  state.tonTraHangList = action.payload;
})
.addCase(getTonTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(getTraHangCTByMaTraHang.pending, (state) => {
  state.loading = true;
})
.addCase(getTraHangCTByMaTraHang.fulfilled, (state, action) => {
  state.loading = false;
  state.TraHangCTList = action.payload;
  console.log("action.payload >> ", action.payload);
})
.addCase(getTraHangCTByMaTraHang.rejected, (state, action) => {
  state.loading = false;
})
.addCase(addNewTraHangCT.pending, (state) => {
```

```
        state.loading = true;
      })
      .addCase(addNewTraHangCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewTraHangCT.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateTraHangCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateTraHangCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateTraHangCT.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteTraHangCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteTraHangCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteTraHangCT.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setHisInfo, setTraHangCTList } =
  TraHangSlice.actions;

export default TraHangSlice.reducer;
```