```
import { createSlice, createAsyncThunk } from "@reduxjs/toolkit";
import { cloneDeep } from "lodash";
import BanHangService from "services/sale/BanHangService";
import { initBanHangCt } from "views/app-views/sale/managements/BanHang/BanLe";

export const getDSBanHang = createAsyncThunk(
  "banhang/getDSBanHang",
  async (data, { rejectWithValue }) => {
    try {
      const response = await BanHangService.searchGrid(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getBanHangByMabanhang = createAsyncThunk(
  "banhang/getBanHangByMamabanhang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await BanHangService.getBanHangByMabanhang(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
  // async (data, { rejectWithValue }) => {
  //   try {
  //     const response = await BanHangService.getBanHangByMabanhang(data);
  //     return response.data;
  //   } catch (err) {
  //     return rejectWithValue(err.message || "Error");
  //   }
  // }
);
export const createBanHang = createAsyncThunk(
  "banhang/createBanHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await BanHangService.create(payload);
      if (onSuccess) onSuccess(response.data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

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

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

export const deleteBanHang = createAsyncThunk(
  "banhang/deleteBanHang",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess, id } = data;
      const response = await BanHangService.deleteBanHang(id);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const getBanHangCTByMabanhang = createAsyncThunk(
  "banhang/getBanHangCTByMabanhang",
  async (data, { rejectWithValue }) => {
    try {
      const response = await BanHangService.getBanHangCTByMabanhang(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const getTonBanHang = createAsyncThunk(
  "banhang/getTonBanHang",
  async (data, { rejectWithValue }) => {
    try {
      const response = await BanHangService.getTonBanHang(data);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

export const addNewBanHangCT = createAsyncThunk(
  "banhang/addNewBanHangCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
      delete payload.onSuccess;
      const response = await BanHangService.addBanHangCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const updateBanHangCT = createAsyncThunk(
  "banhang/updateBanHangCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const payload = cloneDeep(data);
```

```
      delete payload.onSuccess;
      const response = await BanHangService.updateBanHangCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);
export const deleteBanHangCT = createAsyncThunk(
  "banhang/deleteBanHangCT",
  async (data, { rejectWithValue }) => {
    try {
      const { onSuccess } = data;
      const response = await BanHangService.deleteBanHangCT(data);
      if (onSuccess) onSuccess(response);
      return response.data;
    } catch (err) {
      return rejectWithValue(err.message || "Error");
    }
  }
);

const initialState = {
  loading: false,
  banHangList: [],
  banHangDetail: {},
  banHangCTList: [],
  tonBanHangList: [],
};

export const banhangSlice = createSlice({
  name: "banhang",
  initialState,
  reducers: {
    showLoading: (state) => {
      state.loading = true;
    },
    setHisInfo: (state, action) => {
      state.hisInfoList = action.payload;
    },
    setBanHangCTList: (state, action) => {
      state.banHangCTList = action.payload;
    },
  },
  extraReducers: (builder) => {
    builder
      .addCase(getDSBanHang.pending, (state) => {
        state.loading = true;
      })
      .addCase(getDSBanHang.fulfilled, (state, action) => {
        state.loading = false;
        state.banHangList = action.payload;
      })
      .addCase(getDSBanHang.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getBanHangByMabanhang.pending, (state) => {
        state.loading = true;
      })
      .addCase(getBanHangByMabanhang.fulfilled, (state, action) => {
        state.loading = false;
        state.banHangDetail = action.payload;
      })
      .addCase(getBanHangByMabanhang.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(createBanHang.pending, (state) => {
        state.loading = true;
      })
      .addCase(createBanHang.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(createBanHang.rejected, (state, action) => {
```

```
        state.loading = false;
      })
      .addCase(updateBanHang.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateBanHang.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateBanHang.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteBanHang.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteBanHang.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteBanHang.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getTonBanHang.pending, (state) => {
        state.loading = true;
      })
      .addCase(getTonBanHang.fulfilled, (state, action) => {
        state.loading = false;
        state.tonBanHangList = action.payload;
      })
      .addCase(getTonBanHang.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(getBanHangCTByMabanhang.pending, (state) => {
        state.loading = true;
      })
      .addCase(getBanHangCTByMabanhang.fulfilled, (state, action) => {
        state.loading = false;
        state.banHangCTList = [initBanHangCt, ...action.payload];
      })
      .addCase(getBanHangCTByMabanhang.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewBanHangCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(addNewBanHangCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(addNewBanHangCT.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(updateBanHangCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(updateBanHangCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(updateBanHangCT.rejected, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteBanHangCT.pending, (state) => {
        state.loading = true;
      })
      .addCase(deleteBanHangCT.fulfilled, (state, action) => {
        state.loading = false;
      })
      .addCase(deleteBanHangCT.rejected, (state, action) => {
        state.loading = false;
      });
  },
});

export const { showLoading, setHisInfo, setBanHangCTList } =
  banhangSlice.actions;

export default banhangSlice.reducer;
```