Для функционирования иерархии в разделе был создан отдельный миксин, содержащий в себе все необходимые функции для работоспособности иерархии в разделе.

Код миксина будет представлен ниже.

1. В объекте ConfItem добавляем колонку ParentConfItem со ссылкой на самого себя, публикуемся.
2. В странице раздела модифицируем дифф DataGrid следующим образом:

|  |
| --- |
| {      "operation": "merge",      "name": "DataGrid",      "values": {          "hierarchical": true,          "hierarchicalColumnName": "ParentConfItem",          "updateExpandHierarchyLevels": {              "bindTo": "onExpandHierarchyLevels"          },          expandHierarchyLevels: {              "bindTo": "expandHierarchyLevels"          }      }  }, |

1. В моем случае, необходимо было повесить на строку реестра возможность добавления подчиненных записей двух типов. Это было реализовано следующим образом:
   1. В diff добавляем следующий код:

|  |
| --- |
| {      "operation": "insert",      "name": "DataGridActiveRowAddParentRecord",      "parentName": "DataGrid",      "propertyName": "activeRowActions",      "values": {          "className": "Terrasoft.Button",          "caption": resources.localizableStrings.AddChildButtonCaption,          "menu": {              "items": [                  {                      "tag": "addChildRecordMaterial",                      "caption": resources.localizableStrings.MaterialConfItemAddButtonCaption                  },                  {                      "tag": "addChildRecordIntangible",                      "caption": resources.localizableStrings.IntangibleConfItemAddButtonCaption                  }              ]          },          "style": Terrasoft.controls.ButtonEnums.style.GREY,          "visible": true      }  } |

* 1. Добавлем в выборку колонку Parent:

|  |
| --- |
| /\*\*   \* Добавляет в запрос колонку родителя для построения иерархии   \* @param esq экземпляр запроса на выборку данных   \*/  putParentColumn: function (esq) {      if (!esq.columns.contains("ParentConfItem")) {          esq.addColumn("ParentConfItem");      }  },    /\*\*   \* Добавляет дополнительные колонки в запрос на выборку   \* @param esq   \*/  addGridDataColumns: function (esq) {      this.callParent(arguments);      this.putParentColumn(esq);  } |

* 1. замещаем метод onActiveRowChanged:

|  |
| --- |
| onActiveRowAction: function (buttonTag, primaryColumnValue) {      this.callParent(arguments);      switch (buttonTag) {          case "addChildRecordMaterial":          {              this.prepareChildRecordConfigMaterial();              break;          }          case "addChildRecordIntangible":          {              this.prepareChildRecordConfigIntangible();              break;          }      }  }, |

* 1. Добавляем методы для добавления подчиненной записи:

|  |
| --- |
| /\*\*   \* подготавливает тип для открытия карточки   \*/  prepareChildRecordConfigMaterial: function () {      var typeId = TelecomConstants.ConfItemClasses.Material;      this.addChildRecord(typeId);  },    /\*\*   \* подготавливает тип для открытия карточки   \*/  prepareChildRecordConfigIntangible: function () {      var typeId = TelecomConstants.ConfItemClasses.Intangible;      this.addChildRecord(typeId);  },    /\*\*   \* Открывает страницу добавления новой записи с определенным типом   \* @param typeColumnValue   \*/  addChildRecord: function(typeColumnValue) {      var schemaName = this.getEditPageSchemaName(typeColumnValue);      if (!schemaName) {          return;      }      var selectedItem = this.get("ActiveRow");      var typeColumnName = this.get("TypeColumnName");      var name = ["ParentConfItem"];      var value = [selectedItem];      this.sandbox.subscribe("getCardInfo", function() {          return {              typeColumnName: typeColumnName,              typeUId: typeColumnValue,              valuePairs: [{                  name: name,                  value: value              }]          };      }, [this.getChainCardModuleSandboxId(typeColumnValue)]);      var expandedElements = this.get("expandedElements");      this.set("ExpandedLevels", expandedElements);      this.openCardInChain({          schemaName: schemaName,          operation: ConfigurationEnums.CardStateV2.ADD,          moduleId: this.getChainCardModuleSandboxId(typeColumnValue),          valuePairs: [{              name: ["ParentConfItem"],              value: [selectedItem]          }]      });  }, |

* 1. Переопределяем метод initCardModuleResponseSubscription. Это необходимо для корректной перерисовки грида после закрытия карточки:

|  |
| --- |
| /\*\*   \* Обработчик ответа после закрытия страницы.   \* Перезагружает иерархию в соответствии с добавленным значением.   \*/  initCardModuleResponseSubscription: function() {      var editCardsSandboxIds = [];      var editPages = this.get("EditPages");      editPages.each(function(editPage) {          var editCardsSandboxId = this.getChainCardModuleSandboxId(editPage.get("Tag"));          editCardsSandboxIds.push(editCardsSandboxId);      }, this);      editCardsSandboxIds.push(this.getCardModuleSandboxId());      this.sandbox.subscribe("CardModuleResponse", function(response) {          var gridData = this.getGridData();          var expandedElements = this.get("ExpandedLevels");          if (expandedElements){              if (!expandedElements.hasOwnProperty(response.primaryColumnValue)) {                  expandedElements[response.primaryColumnValue] = {                      page: 0                  };              }              this.set("expandedElements", expandedElements);              this.setExpandedItem(response.primaryColumnValue);              this.set("ExpandItemId", response.primaryColumnValue);              this.loadGridData();              this.loadGridDataRecord(response.primaryColumnValue);              this.reloadGridData();          } else {              gridData.clear();              this.loadGridData();          }      }, this, editCardsSandboxIds);    }, |

1. Подключаем миксин:

|  |
| --- |
| define("HierarchicalSectionUtilities", ["HierarchicalSectionUtilitiesResources", "GridUtilitiesV2"],      function(resources) {          Ext.define("Terrasoft.configuration.mixins.HierarchicalSectionUtilities", {              extend: "Terrasoft.BaseObject",              alternateClassName: "Terrasoft.HierarchicalSectionUtilities",                expandedElements: {},              expandHierarchyLevels: {},              ExpandItemId: null,              ExpandedLevels: null,                changeSorting: function() {                  this.clearExpandHierarchyLevels();                  this.callParent(arguments);                },                /\*\*               \* Обработчик загрузки дочерних элементов. Осуществляет проверку загружености               \* дочерних элеметов выбранной записи. Запускает загрузку нового уровня.               \* @protected               \* @virtual               \* @param {String} primaryColumnValue Уникальный идентификатор записи               \* @param {Boolean} isExpanded Признак того, разворачивает или сворачивает пользователь дочерние элеметы               \* true - если разворачивает, false в обратном случае               \*/              onExpandHierarchyLevels: function(primaryColumnValue, isExpanded) {                  if (!isExpanded || this.isItemExpanded(primaryColumnValue)) {                      return;                  }                  this.setExpandedItem(primaryColumnValue);                  this.set("ExpandItemId", primaryColumnValue);              },                /\*\*               \* Удаляет логику постраничности если загружаются дочерние объекты               \* @protected               \* @overridden               \*/              initCanLoadMoreData: function() {                  var parentItem = this.get("ExpandItemId");                  if (!parentItem) {                      this.callParent(arguments);                  }              },                /\*\*               \* получает коллекцию развернутых уровней               \* @returns {\*}               \*/              getExpandedItems: function() {                  return this.get("expandedElements");              },                /\*\*               \* Устанавливает новый уровень развернутости иерархии               \* @param primaryColumnValue               \*/              setExpandedItem: function(primaryColumnValue) {                  (this.getExpandedItems()[primaryColumnValue]) = { "page": 0 };              },                /\*\*               \* Проверяет, были ли загруженны дочерные элементы выбранной записи               \* @protected               \* @virtual               \* @param primaryColumnValue Уникальный идетификатор записи               \* @return {boolean}               \*/              isItemExpanded: function(primaryColumnValue) {                  return !!(this.getExpandedItems()[primaryColumnValue]);              },                /\*\*               \* Очищет информацию о загруженных и развернутых уровнях               \* @protected               \* @virtual               \*/              clearExpandHierarchyLevels: function() {                  this.set("expandedElements", {});                  this.set("expandHierarchyLevels", []);                  var grid = this.getCurrentGrid();                  if (grid) {                      grid.expandHierarchyLevels = [];                  }              },                /\*\*               \* Убирает информацию о том, что элемент развернут из системных параметров               \* @protected               \* @virtual               \* @param {String} itemId Уникальный идентификатор записи               \*/              removeExpandHierarchyLevel: function(itemId) {                  var expandHierarchyLevels = this.get("expandHierarchyLevels");                  this.set("expandHierarchyLevels", Terrasoft.without(expandHierarchyLevels, itemId));                  var grid = this.getCurrentGrid();                  if (grid) {                      grid.expandHierarchyLevels = Terrasoft.without(grid.expandHierarchyLevels, itemId);                  }              },                /\*\*               \* Дополнительная обработка после удаления записи               \* Очищает иерархию и перезагружает реестр               \* @overridden               \* @protected               \*/              onDeleted: function(result) {                  this.callParent(arguments);                    if (result.Success) {                      this.clearExpandHierarchyLevels();                      this.loadGridData();                      this.reloadGridData();                  }              },                reloadGridColumnsConfig: function(doReRender) {                  var profile = this.get("Profile");                  var propertyName = this.getDataGridName();                  var gridProfile = profile[propertyName];                  if (!this.Ext.Object.isEmpty(gridProfile)) {  /\*\*  \* очень важны две следующие строки, благодаря им при перерисовке реестра в виде вертикального, не разрушаются стили иерархии, и она нормально отображается в вертикальном реестре.  \*/                      gridProfile.isTiled = false;                      gridProfile.type = "listed";                      var grid = this.getCurrentGrid();                      if (!grid) {                          return;                      }                      grid.type = gridProfile.type;                      var viewGenerator = Ext.create("Terrasoft.ViewGenerator");                      viewGenerator.viewModelClass = this;                      var gridConfig;                      var bindings = this.Terrasoft.deepClone(grid.bindings);                      if (gridProfile.type === Terrasoft.GridType.LISTED) {                          gridConfig = {                              listedConfig: Ext.decode(gridProfile.listedConfig),                              type: gridProfile.type                          };                          viewGenerator.actualizeListedGridConfig(gridConfig);                          grid.captionsConfig = gridConfig.listedConfig.captionsConfig;                          grid.columnsConfig = gridConfig.listedConfig.columnsConfig;                          grid.listedConfig = gridConfig.listedConfig;                          grid.initBindings(gridConfig.listedConfig);                      } else {                          gridConfig = {                              tiledConfig: Ext.decode(gridProfile.tiledConfig),                              type: gridProfile.type                          };                          var isVertical = (propertyName.indexOf("VerticalProfile") > -1);                          gridConfig.isVertical = isVertical;                          grid.isEmptyRowVisible = !isVertical;                          viewGenerator.actualizeTiledGridConfig(gridConfig);                          grid.columnsConfig = gridConfig.tiledConfig.columnsConfig;                          grid.tiledConfig = gridConfig.tiledConfig;                          grid.initBindings(gridConfig.tiledConfig);                      }                      grid.bindings = bindings;                      if (doReRender) {                          grid.clear();                          grid.prepareCollectionData();                          if (grid.rendered) {                              grid.reRender();                          }                      }                  }              }          });          return Terrasoft.HierarchicalSectionUtilities;      }  ); |

1. Создаем модуль стилей и подключаем его в вашу секцию (это нужно для того чтобы плюсики иерархии не уезжали слишком далеко. Код стилей:

|  |
| --- |
| //здесь можно реализовать один и тот же стиль для разных разделов, где нужно настраивать иерархию  #ProductsSectionGridDataViewContainer .grid-fixed-col,  #ConfItemSectionGridDataViewContainer .grid-fixed-col,  #ProductCatalogueLvlSectionV2GridDataViewContainer .grid-fixed-col  {      padding-left: 24px;  } |

1. Иерархия готова.