Load On First:

```
<CTreeView @ref="expandedTree" TEntity="TreeViewItem" OnChange="ExpandedNodeChange" Source="expandedSource" />
```

Collapsed:

```
<CTreeView @ref="tree1" OnChange="CollapsedNodeChange" OnCollapsed="OnCollapsed" TEntity="TreeViewItem"
      Source="collapsedSource" OnExpanded="async node => await OnExpanded(node)" />
```

```
@code
{
    bool expanded;
    int expandedSeletedNodesCount;
    int collapsedSeletedNodesCount;
    IList<TreeViewItem> expandedSource;
    IList<TreeViewItem> collapsedSource;
    CTreeView<TreeViewItem> expandedTree;
    CTreeView<TreeViewItem> tree1;

    protected override async Task OnInitializedAsync()
    {
        using var scope = CreateScope();
        expandedSource = await new MenuCategoryService(scope.ServiceProvider).GetAll().Where(t => t.SubSystemKind ==
             SubSystemKind.Demo)
            .Select(t => new TreeViewItem()
                {
                    Text = t.Title,
                    Expanded = false,
                    Selectable = true,
                    Value = t.Id.ToString(),
                    Items = t.Menus.Select(u => new TreeViewItem()
                    {
                        Collabsable = false,
                        Text = u.Title,
                        Value = u.Id.ToString(),
                        Selectable = true
                    }).ToList()
                }).ToListAsync();
        collapsedSource = expandedSource.Select(t => new TreeViewItem()
        {
            Expanded = false,
            Collabsable = true,
            Selectable = true,
            Text = t.Text,
            Value = t.Value
        }).ToList();
        await base.OnInitializedAsync();
    }

    void OnCollapsed(TreeViewItem node)
    {
        node.Items = null;
        node.Selected = false;

    }

    void ExpandedNodeChange(TreeViewItem item)
    {
        // This is for tow level tree
        if (item.Items != null)
        {
            foreach (var child in item.Items)
                child.Selected = item.Selected;
        }
        var parent = item.Parent;
        if (parent != null)
        {
            var siblings = parent.Items;
            if (siblings.All(t => t.Selected == true))
                parent.Selected = true;
            else if (siblings.All(t => t.Selected == false))
                parent.Selected = false;
            else
                parent.Selected = null;
            parent = parent.Parent;
        }
        expandedSeletedNodesCount = expandedTree.GetSeletcedItems().Count(t => t.Depth == 2);
    }
```

```
void CollapsedNodeChange(TreeViewItem node)
{
    if (node.Depth == 1)
    {
        if (node.Items == null)
            node.Selected = false;
        else
        {
            foreach (var item in node.Items)
                item.Selected = node.Selected;
        }
    }
    if (node.Depth == 2)
    {
        var parent = node.Parent;
        var items = parent.Items;
        if (items.All(t => t.Selected == true))
            parent.Selected = true;
        else if (items.All(t => t.Selected == false))
            parent.Selected = false;
        else
            parent.Selected = null;
    }
    collapsedSeletedNodesCount = tree1.GetSeletcedItems().Count(t => t.Depth == 2);
}

async Task OnExpanded(TreeViewItem node)
{
    using var scope = CreateScope();
        var categoryId = Convert.ToInt32(node.Value);
        node.Items = await new MenuService(scope.ServiceProvider).GetAll()
            .Where(t => t.MenuCategoryId == categoryId).Select(t => new TreeViewItem()
                {
                    Collabsable = false,
                    Expanded = false,
                    Selectable = true,
                    Text = t.Title,
                    Value = t.Id.ToString()
                }).ToListAsync();
    }

    void ExpandedChanged()
    {
        foreach (var item in expandedSource)
            item.Expanded = expanded;
    }
}
```