**MatTree** is a useful component for displaying hierarchical data in a tree-like structure. It can be configured to have multiple levels of nodes, each with a label and an optional icon. MatTree also supports keyboard navigation, accessibility features, and drag and drop functionality.

**MatTable** is a useful component for displaying tabular data in a grid-like structure. It can be configured to have multiple columns, each with a header and a cell template. MatTable also supports sorting, filtering, pagination, and selection functionality.

**MatTABLE** with **MatTREE** is hybrid solution for Table to have Nested child functionality along with support of Mat Pagination, Mat Sort and Search Filter.

In order to achieve this component, there are couple of additional methods are required to be added what already provided by Angular Material for Tree and Mat Table.

You may look at the example code here, which have those additional methods. This document would summarize the methods as: -

* **FlatTreeControl:** This class is used to control a flat tree, i.e., a tree with an underlying flat list of nodes. Some of its methods include:
  1. expand: Expands a given data node in the tree.
  2. collapse: Collapses a given data node in the tree.
  3. getLevel: Gets the level of a given tree node.
* **MatTreeFlatDataSource:** This class is a data source for a flat list that is used in conjunction with MatTreeFlattener. Some of its methods include:

1. connect: Connects a collection viewer (such as a MatTree) to this data source.
2. disconnect: Disconnects a collection viewer from this data source.

* **MatTreeFlattener:** This class is used to convert a normal tree (i.e., a hierarchical tree structure) into a flat tree. Some of its methods include:

1. flattenNode: Flattens a particular node in the tree.
2. expandLevel: Expands the level of a particular node in the tree.

* **MatPaginator:** This class provides a paginator for a MatTable. Some of its methods include:

1. nextPage: Advances to the next page.
2. previousPage: Moves to the previous page.
3. firstPage: Moves to the first page.
4. lastPage: Moves to the last page.

* **MatSort:** This class provides sorting functionality for a MatTable. Some of its methods include:

1. sort: Sorts the data based on a sort header id and a direction (asc or desc).

The **FoodFlatNode** class currently does not have any methods. It only has properties name, count, children, and text.

Here's a brief explanation of the methods in this class:

* **constructor():** It initializes the dataSource with the buildFileTree method and sets the pageSize.
* **ngOnInit():** Here, it sets the length and page size of the paginator, and sorts the columns if sorting is provided.
* **hasChild =** (\_: number, node: FoodFlatNode) => node.expandable;: This method checks if a node is expandable.
* **buildFileTree(obj: any[], level: string): FoodFlatNode[]:** This method builds the file structure tree. It takes an array of objects and a level as parameters and returns an array of FoodFlatNode.
* **onPageChange(event: MatPaginator | any):** This method handles page changes in the paginator. It updates the pageSize and the dataSource.data.
* **sortHeaders**(col: string, order?: string): This method sorts the headers based on the column and order provided. It sorts the data in ascending or descending order.
* **getColumns**(col: string): This method returns the column with the given id.
* **filterChanged**(event: any): This method is called when the filter changes. It filters the data and expands or collapses all nodes based on the filter text.
* **filter(filterText: string):** This method filters the data based on the filter text provided. It builds the tree nodes from the filtered data and notifies the change.

Please note that this is a high-level explanation. The actual behaviour of these methods depends on the data they are working with and how they are used in the component's template.