# 2nd Fase EDD

1.0

Generated by Doxygen 1.9.1

# Chapter 1

# Modules Index

## 1.1 Modules List

Here is a list of all modules with brief descriptions:

# Chapter 2

# Data Type Index

## 2.1 Data Types List

Here are the data types with brief descriptions:

# Chapter 3

# File Index

## 3.1 File List

Here is a list of all files with brief descriptions:

# Chapter 4

# Module Documentation

## 4.1 albums Module Reference

### Data Types

- type img_node

  *Type to store the image node.*
- type album

  *Type to store the album.*
- type album_list

  *Type to store the album list.*

### Functions/Subroutines

- subroutine gen_album_graph (this, unit)

  *Generates the graph of the albums.*
- subroutine show_album_images (this, id_album)

  *Shows the images of an album.*
- subroutine show_albums (this)

  *Shows the albums.*
- subroutine remove_image_from_album (this, ralbum_id, image_id)

  *Removes an image from an album.*
- subroutine add_image_to_album (this, ialbum_id, image_node)

  *Adds an image to an album.*
- type(image) function, pointer search_in_album (this, salbum_id, image_id)

  *Searches an image in an album.*
- type(album) function, pointer get_album (this, id)

  *Gets an album.*
- subroutine remove_album (this, id)

  *Removes an album.*
- subroutine new_album (this, album_name)

  *Creates a new album.*
- subroutine add_image (this, image_node)

  *Adds an image to an album.*
- subroutine remove_image (this, id)

  *Removes an image from an album.*
- type(image) function, pointer get_image (this, id)

  *Gets an image.*
- subroutine show_images (this)

  *Shows the images of an album.*

## Variables

- integer album_id = 0

  *Global variable to store the album id.*

### 4.1.1 Function/Subroutine Documentation

#### 4.1.1.1 add_image()

```
subroutine albums::add_image (
             class(album), intent(inout) this,
             type(image), intent(in), pointer image_node )
```

Adds an image to an album.

**Parameters**

| | |
|---|---|
| *this* | Album |
| *image_node* | Image node |

Definition at line 253 of file albums.f90.

#### 4.1.1.2 add_image_to_album()

```
subroutine albums::add_image_to_album (
             class(album_list), intent(inout) this,
             integer, intent(in) ialbum_id,
             type(image), intent(in), pointer image_node )
```

Adds an image to an album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *ialbum_id* | Album id |
| *image_node* | Image node |

Definition at line 139 of file albums.f90.

#### 4.1.1.3 gen_album_graph()

```
subroutine albums::gen_album_graph (
```

```
            class(album_list), intent(in) this,
            integer, intent(in) unit )
```

Generates the graph of the albums.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *unit* | Unit to write the graph |

Definition at line 52 of file albums.f90.

#### 4.1.1.4 get_album()

```
type(album) function, pointer albums::get_album (
            class(album_list), intent(in) this,
            integer, intent(in) id )
```

Gets an album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *id* | Album id |

**Returns**

Album node

Definition at line 173 of file albums.f90.

#### 4.1.1.5 get_image()

```
type(image) function, pointer albums::get_image (
            class(album), intent(in) this,
            integer, intent(in) id )
```

Gets an image.

**Parameters**

| | |
|---|---|
| *this* | Album |
| *id* | Image id |

**Returns**

Image node

Definition at line 310 of file albums.f90.

#### 4.1.1.6 new_album()

```
subroutine albums::new_album (
             class(album_list), intent(inout) this,
             character(*), intent(in) album_name )
```

Creates a new album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *album_name* | Album name |

Definition at line 232 of file albums.f90.

#### 4.1.1.7 remove_album()

```
subroutine albums::remove_album (
             class(album_list), intent(inout) this,
             integer, intent(in) id )
```

Removes an album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *id* | Album id |

Definition at line 194 of file albums.f90.

#### 4.1.1.8 remove_image()

```
subroutine albums::remove_image (
             class(album), intent(inout) this,
             integer, intent(in) id )
```

Removes an image from an album.

**Parameters**

| | |
|---|---|
| *this* | Album |
| *id* | Image id |

Definition at line 272 of file albums.f90.

#### 4.1.1.9 remove_image_from_album()

```
subroutine albums::remove_image_from_album (
             class(album_list), intent(inout) this,
             integer, intent(in) ralbum_id,
             integer, intent(in) image_id )
```

Removes an image from an album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *ralbum_id* | Album id |
| *image_id* | Image id |

Definition at line 123 of file albums.f90.

#### 4.1.1.10 search_in_album()

```
type(image) function, pointer albums::search_in_album (
             class(album_list), intent(in) this,
             integer, intent(in) salbum_id,
             integer, intent(in) image_id )
```

Searches an image in an album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *salbum_id* | Album id |
| *image_id* | Image id |

**Returns**

Image node

Definition at line 156 of file albums.f90.

#### 4.1.1.11 show_album_images()

```
subroutine albums::show_album_images (
            class(album_list), intent(in) this,
            integer, intent(in) id_album )
```

Shows the images of an album.

**Parameters**

| | |
|---|---|
| *this* | Album list |
| *id_album* | Album id |

Definition at line 92 of file albums.f90.

#### 4.1.1.12 show_albums()

```
subroutine albums::show_albums (
            class(album_list), intent(in) this )
```

Shows the albums.

**Parameters**

| | |
|---|---|
| *this* | Album list |

Definition at line 106 of file albums.f90.

#### 4.1.1.13 show_images()

```
subroutine albums::show_images (
            class(album), intent(in) this )
```

Shows the images of an album.

**Parameters**

| | |
|---|---|
| *this* | Album |

Definition at line 332 of file albums.f90.

### 4.1.2 Variable Documentation

#### 4.1.2.1 album_id

```
integer albums::album_id = 0
```

Global variable to store the album id.

Definition at line 10 of file albums.f90.

## 4.2 clients Module Reference

### Data Types

- type client
  *Type to represent a client.*
- type nodeptr
  *Type to represent a node pointer.*
- type btreenode
  *Type to represent a B-tree node.*
- type btree_clients
  *Type to represent a B-tree.*
- type q_node
  *Type to represent a queue node.*
- type client_queue
  *Type to represent a client queue.*

### Functions/Subroutines

- subroutine amplitude_traversal (this)
  *Subroutine to traverse the tree in amplitude.*
- logical function is_empty (this)
  *Function to check if the queue is empty.*
- type(btreenode) function, pointer dequeue (this)
  *Function to dequeue a node from the queue.*
- subroutine enqueue (this, tree_node)
  *Subroutine to enqueue a node in the queue.*
- subroutine add_client (this, new_client)
  *Subroutine to add a client to the tree.*
- recursive logical function set_value (new_client, pclient, node, child)
  *Recursive function to set the value of the tree.*
- type(btreenode) function, pointer create_node (this, new_client, child)
  *Function to create a node.*
- subroutine insert_node (pclient, pos, node, child)
  *Subroutine to insert a node.*
- subroutine split_node (new_client, pclient, pos, node, child, new_node)

*Subroutine to split a node.*

- subroutine traversal (this, node)

  *Subroutine to traverse the tree.*
- subroutine clients_dot (this, node, unit)

  *Subroutine to print the tree in dot format.*
- recursive type(client) function, pointer search_client (this, node, client_id)

  *Function to search a client in the tree.*
- subroutine delete_client (this, client_id)

  *Subroutine to delete a client from the tree.*
- subroutine delete_client_rec (this, father, temp, client_id)

  *Recursive subroutine to delete a client from the tree.*
- logical function find (this, client_id)

  *Function to find a client in the node.*

## Variables

- integer g_id = 1

  *Global variable to assign the id of the nodes.*

### 4.2.1 Function/Subroutine Documentation

#### 4.2.1.1 add_client()

```
subroutine clients::add_client (
             class(btree_clients), intent(inout) this,
             type(client), intent(in) new_client )
```

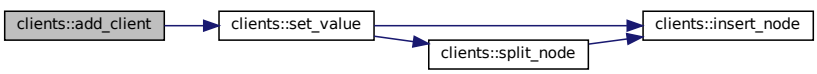
Subroutine to add a client to the tree.

**Parameters**

| | |
|---|---|
| *this* | B-tree to add the client |
| *new_client* | Client to add |

Definition at line 133 of file clients.f90.

Here is the call graph for this function:



#### 4.2.1.2 amplitude_traversal()

```
subroutine clients::amplitude_traversal (
             class(btree_clients), intent(inout) this )
```

Subroutine to traverse the tree in amplitude.

**Parameters**

| | |
|---|---|
| *this* | B-tree to traverse |

Definition at line 65 of file clients.f90.

#### 4.2.1.3 clients_dot()

```
subroutine clients::clients_dot (
             class(btree_clients), intent(inout) this,
             type(btreenode), intent(in), pointer node,
             integer, intent(in) unit )
```

Subroutine to print the tree in dot format.

**Parameters**

| | |
|---|---|
| *this* | B-tree to print |
| *node* | Node to print |
| *unit* | Unit to print |

Definition at line 303 of file clients.f90.

#### 4.2.1.4 create_node()

```
type(btreenode) function, pointer clients::create_node (
             class(btree_clients), intent(inout) this,
             type(client), intent(in) new_client,
             type(btreenode), intent(in), pointer child )
```

Function to create a node.

**Parameters**

| | |
|---|---|
| *this* | B-tree to create the node |
| *new_client* | Client to add |
| *child* | Child node |

**Returns**

New node created

Definition at line 198 of file clients.f90.

#### 4.2.1.5 delete_client()

```
subroutine clients::delete_client (
             class(btree_clients), intent(inout) this,
             integer(kind=8), intent(in) client_id )
```

Subroutine to delete a client from the tree.

**Parameters**

| | |
|---|---|
| *this* | B-tree to delete the client |
| *client_id* | Id of the client to delete |

Definition at line 358 of file clients.f90.

#### 4.2.1.6 delete_client_rec()

```
subroutine clients::delete_client_rec (
             class(btree_clients), intent(inout) this,
             type(btreenode), intent(inout), pointer father,
             type(btreenode), intent(inout), pointer temp,
             integer(kind=8), intent(in) client_id )
```

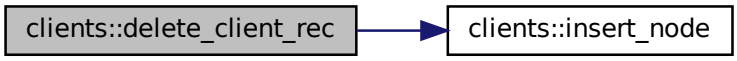
Recursive subroutine to delete a client from the tree.

**Parameters**

| | |
|---|---|
| *this* | B-tree to delete the client |
| *father* | Father node |
| *temp* | Node to delete the client |
| *client_id* | Id of the client to delete |

Definition at line 372 of file clients.f90.

Here is the call graph for this function:




#### 4.2.1.7 dequeue()

```
type(btreenode) function, pointer clients::dequeue (
            class(client_queue), intent(inout) this )
```

Function to dequeue a node from the queue.

**Parameters**

| | |
|---|---|
| *this* | Queue to dequeue |

**Returns**

Node dequeued

Definition at line 97 of file clients.f90.

#### 4.2.1.8 enqueue()

```
subroutine clients::enqueue (
            class(client_queue), intent(inout) this,
            type(btreenode), intent(in), pointer tree_node )
```

Subroutine to enqueue a node in the queue.

**Parameters**

| | |
|---|---|
| *this* | Queue to enqueue |
| *tree_node* | Node to enqueue |

Definition at line 112 of file clients.f90.

#### 4.2.1.9 find()

```
logical function clients::find (
             class(btreenode), intent(in) this,
             integer(kind=8), intent(in) client_id )
```

Function to find a client in the node.

**Parameters**

| | |
|---|---|
| *this* | Node to search |
| *client↩ _id* | Id of the client to search |

**Returns**

True if the client was found, false otherwise

Definition at line 428 of file clients.f90.

#### 4.2.1.10 insert_node()

```
subroutine clients::insert_node (
             type(client), intent(in) pclient,
             integer, intent(in) pos,
             type(btreenode), intent(inout), pointer node,
             type(btreenode), intent(in), pointer child )
```

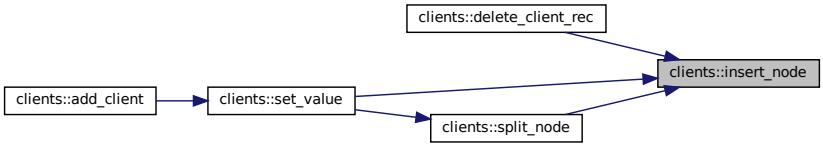
Subroutine to insert a node.

**Parameters**

| | |
|---|---|
| *pclient* | Client to add |
| *pos* | Position to add the client |
| *node* | Node to add the client |
| *child* | Child node |

Definition at line 220 of file clients.f90.

Here is the caller graph for this function:



#### 4.2.1.11 is_empty()

```
logical function clients::is_empty (
            class(client_queue), intent(in) this )
```

Function to check if the queue is empty.

**Parameters**

| | |
|---|---|
| *this* | Queue to check |

**Returns**

True if the queue is empty, false otherwise

Definition at line 89 of file clients.f90.

#### 4.2.1.12 search_client()

```
recursive type(client) function, pointer clients::search_client (
            class(btree_clients), intent(inout) this,
            type(btreenode), intent(in), pointer node,
            integer(kind=8), intent(in) client_id )
```

Function to search a client in the tree.

**Parameters**

| | |
|---|---|
| *this* | B-tree to search |
| *node* | Node to search |
| *client↩ _id* | Id of the client to search |

**Returns**

Client found

Definition at line 331 of file clients.f90.

#### 4.2.1.13 set_value()

```
recursive logical function clients::set_value (
            type(client), intent(in) new_client,
```

```
            type(client), intent(inout) pclient,
            type(btreenode), intent(inout), pointer node,
            type(btreenode), intent(inout), pointer child )
```

Recursive function to set the value of the tree.

**Parameters**

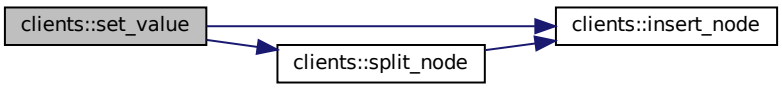
| | |
|---|---|
| *new_client* | Client to add |
| *pclient* | Client to add |
| *node* | Node to add the client |
| *child* | Child node |

**Returns**

True if the client was added, false otherwise

Definition at line 151 of file clients.f90.

Here is the call graph for this function:




Here is the caller graph for this function:


#### 4.2.1.14 split_node()

```
subroutine clients::split_node (
            type(client), intent(in) new_client,
            type(client), intent(inout) pclient,
            integer, intent(in) pos,
            type(btreenode), intent(inout), pointer node,
            type(btreenode), intent(in), pointer child,
            type(btreenode), intent(inout), pointer new_node )
```

Subroutine to split a node.

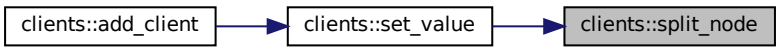
**Parameters**

| | |
|---|---|
| *new_client* | Client to add |
| *pclient* | Client to add |
| *pos* | Position to add the client |
| *node* | Node to split |
| *child* | Child node |
| *new_node* | New node |

Definition at line 243 of file clients.f90.

Here is the call graph for this function:


Here is the caller graph for this function:




#### 4.2.1.15 traversal()

```
subroutine clients::traversal (
             class(btree_clients), intent(inout) this,
             type(btreenode), intent(in), pointer node )
```

Subroutine to traverse the tree.

**Parameters**

| | |
|---|---|
| *this* | B-tree to traverse |
| *node* | Node to traverse |

Definition at line 281 of file clients.f90.

### 4.2.2 Variable Documentation

#### 4.2.2.1 g_id

```
integer clients::g_id = 1
```

Global variable to assign the id of the nodes.

Definition at line 13 of file clients.f90.

## 4.3 filehandler Module Reference

### Data Types

- type fhandler

  *Type that handles the reading of the json files and the initialization of the data structures.*

### Functions/Subroutines

- subroutine initialize_admin (this)

  *Initializes the data structures for the admin user.*
- subroutine initialize_user (this)

  *Initializes the data structures for the user.*
- subroutine set_user (this, user)

  *Sets the user data structures to the fhandler object.*
- subroutine read_imgs (this)

  *Reads the images from the json file.*
- subroutine read_layers (this)

  *Reads the layers from the json file.*
- subroutine read_clients (this)

  *Reads the clients from the json file.*
- subroutine read_albums (this)

  *Reads the albums from the json file.*

### 4.3.1 Function/Subroutine Documentation

#### 4.3.1.1 initialize_admin()

```
subroutine filehandler::initialize_admin (
             class(fhandler), intent(inout) this )
```

Initializes the data structures for the admin user.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |

Definition at line 34 of file filehandler.f90.

#### 4.3.1.2 initialize_user()

```
subroutine filehandler::initialize_user (
             class(fhandler), intent(inout) this )
```

Initializes the data structures for the user.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |

Definition at line 40 of file filehandler.f90.

#### 4.3.1.3 read_albums()

```
subroutine filehandler::read_albums (
             class(fhandler), intent(inout) this )
```

Reads the albums from the json file.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |

Definition at line 189 of file filehandler.f90.

#### 4.3.1.4 read_clients()

```
subroutine filehandler::read_clients (
             class(fhandler), intent(inout) this )
```

Reads the clients from the json file.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |

Definition at line 153 of file filehandler.f90.

#### 4.3.1.5 read_imgs()

```
subroutine filehandler::read_imgs (
             class(fhandler), intent(inout) this )
```

Reads the images from the json file.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |

Definition at line 59 of file filehandler.f90.

#### 4.3.1.6 read_layers()

```
subroutine filehandler::read_layers (
             class(fhandler), intent(inout) this )
```

Reads the layers from the json file.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |

Definition at line 100 of file filehandler.f90.

#### 4.3.1.7 set_user()

```
subroutine filehandler::set_user (
             class(fhandler), intent(inout) this,
             type(client), intent(in), pointer user )
```

Sets the user data structures to the fhandler object.

**Parameters**

| | |
|---|---|
| *this* | The fhandler object |
| *user* | The user object |

Definition at line 50 of file filehandler.f90.

## 4.4 images Module Reference

### Data Types

- type image

  *Type that represents a node in the AVL tree.*
- type image_avl

  *Type that represents an AVL tree of images.*

### Functions/Subroutines

- subroutine delete_img (this, img_id)

  *Method that deletes an image from the tree.*
- recursive type(image) function, pointer delete_img_rec (this, temp, img_id)

  *Recursive method that deletes an image from the tree.*
- recursive type(image) function, pointer min_child (this, temp)

  *Method that returns the node with the minimum value in the tree.*
- subroutine add_layer (this, new_layer)

  *Method that adds a layer to the image.*
- subroutine add_img (this, new_image)

  *Method that adds an image to the tree.*
- subroutine add_img_rec (this, new_image, tmp)

  *Recursive method that adds an image to the tree.*
- recursive type(image) function, pointer search_img (this, temp, img_ig)

  *Method that searches for an image in the tree.*
- type(image) function, pointer srl (this, t1)

  *Method that performs a single right rotation.*
- type(image) function, pointer srr (this, t1)

  *Method that performs a single left rotation.*
- type(image) function, pointer drl (this, tmp)

  *Method that performs a double right rotation.*
- type(image) function, pointer drr (this, tmp)

  *Method that performs a double left rotation.*
- integer function get_max (this, val1, val2)

  *Method that returns the maximum value between two integers.*
- integer function get_height (this, tmp)

  *Method that returns the height of a node.*
- subroutine get_dot (this, tmp, unit)

  *Method that generates the dot representation of the tree.*
- subroutine get_dot_rec (this, tmp, unit)

  *Recursive method that generates the dot representation of the tree.*
- subroutine print_images (this, temp)

  *Method that prints the images in the tree.*
- subroutine gen_tree_subtree (this, id_img, unit)

  *Method that generates the dot representation of the tree and its subtree.*
- recursive subroutine gen_layer_subtree (current_layer, unit)

  *Recursive method that generates the dot representation of the layers subtree.*
- subroutine gen_img_traversal (this, unit, id_img)

  *Method that generates the dot representation of the image traversal.*

### 4.4.1 Function/Subroutine Documentation

#### 4.4.1.1 add_img()

```
subroutine images::add_img (
             class(image_avl), intent(inout) this,
             type(image), intent(in), pointer new_image )
```

Method that adds an image to the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *new_image* | The image to add |

Definition at line 145 of file images.f90.

#### 4.4.1.2 add_img_rec()

```
subroutine images::add_img_rec (
             class(image_avl), intent(inout) this,
             type(image), intent(in), pointer new_image,
             type(image), pointer tmp )
```

Recursive method that adds an image to the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *new_image* | The image to add |
| *tmp* | The current node |

Definition at line 159 of file images.f90.

#### 4.4.1.3 add_layer()

```
subroutine images::add_layer (
             class(image), intent(inout) this,
             type(layer), intent(in), pointer new_layer )
```

Method that adds a layer to the image.

**Parameters**

| | |
|---|---|
| *this* | The image object |
| *new_layer* | The layer to add |

Definition at line 136 of file images.f90.

#### 4.4.1.4 delete_img()

```
subroutine images::delete_img (
             class(image_avl), intent(inout) this,
             integer, intent(in) img_id )
```

Method that deletes an image from the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *img_id* | The id of the image to delete |

Definition at line 48 of file images.f90.

#### 4.4.1.5 delete_img_rec()

```
recursive type(image) function, pointer images::delete_img_rec (
             class(image_avl), intent(inout) this,
             type(image), intent(inout), pointer temp,
             integer, intent(in) img_id )
```

Recursive method that deletes an image from the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *temp* | The current node |
| *img_id* | The id of the image to delete |

**Returns**

The new subtree

Definition at line 62 of file images.f90.

#### 4.4.1.6 drl()

```
type(image) function, pointer images::drl (
              class(image_avl), intent(in) this,
              type(image), intent(in), pointer tmp )
```

Method that performs a double right rotation.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *tmp* | The node to rotate |

**Returns**

The new subtree

Definition at line 244 of file images.f90.

#### 4.4.1.7 drr()

```
type(image) function, pointer images::drr (
              class(image_avl), intent(in) this,
              type(image), intent(in), pointer tmp )
```

Method that performs a double left rotation.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *tmp* | The node to rotate |

**Returns**

The new subtree

Definition at line 255 of file images.f90.

#### 4.4.1.8 gen_img_traversal()

```
subroutine images::gen_img_traversal (
              class(image_avl), intent(inout) this,
              integer, intent(in) unit,
              integer, intent(in) id_img )
```

Method that generates the dot representation of the image traversal.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *unit* | The unit to write the dot representation |
| *id_img* | The id of the image to generate the traversal |

Definition at line 400 of file images.f90.

#### 4.4.1.9 gen_layer_subtree()

```
recursive subroutine images::gen_layer_subtree (
             type(layer), intent(in), pointer current_layer,
             integer, intent(in) unit )
```

Recursive method that generates the dot representation of the layers subtree.

**Parameters**

| | |
|---|---|
| *current_layer* | The current layer |
| *unit* | The unit to write the dot representation |

Definition at line 383 of file images.f90.

Here is the caller graph for this function:


#### 4.4.1.10 gen_tree_subtree()

```
subroutine images::gen_tree_subtree (
             class(image_avl), intent(inout) this,
             integer, intent(in) id_img,
             integer, intent(in) unit )
```

Method that generates the dot representation of the tree and its subtree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *id_img* | The id of the image to generate the subtree |
| *unit* | The unit to write the dot representation |

Definition at line 337 of file images.f90.

Here is the call graph for this function:


#### 4.4.1.11 get_dot()

```
subroutine images::get_dot (
             class(image_avl), intent(in) this,
             type(image), intent(in), pointer tmp,
             integer, intent(in) unit )
```

Method that generates the dot representation of the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *tmp* | The current node |
| *unit* | The unit to write the dot representation |

Definition at line 291 of file images.f90.

#### 4.4.1.12 get_dot_rec()

```
subroutine images::get_dot_rec (
             class(image_avl), intent(in) this,
             type(image), intent(in), pointer tmp,
             integer, intent(in) unit )
```

Recursive method that generates the dot representation of the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *tmp* | The current node |
| *unit* | The unit to write the dot representation |

Definition at line 303 of file images.f90.

#### 4.4.1.13 get_height()

```
integer function images::get_height (
            class(image_avl), intent(in) this,
            type(image), intent(in), pointer tmp )
```

Method that returns the height of a node.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *tmp* | The node |

**Returns**

The height of the node

Definition at line 277 of file images.f90.

#### 4.4.1.14 get_max()

```
integer function images::get_max (
            class(image_avl), intent(in) this,
            integer, intent(in) val1,
            integer, intent(in) val2 )
```

Method that returns the maximum value between two integers.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *val1* | The first value |
| *val2* | The second value |

**Returns**

The maximum value

Definition at line 267 of file images.f90.

#### 4.4.1.15 min_child()

```
recursive type(image) function, pointer images::min_child (
            class(image_avl), intent(inout) this,
            type(image), intent(in), pointer temp )
```

Method that returns the node with the minimum value in the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *temp* | The current node |

**Returns**

The node with the minimum value

Definition at line 124 of file images.f90.

#### 4.4.1.16 print_images()

```
subroutine images::print_images (
            class(image_avl), intent(in) this,
            type(image), intent(in), pointer temp )
```

Method that prints the images in the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *temp* | The current node |

Definition at line 323 of file images.f90.

#### 4.4.1.17 search_img()

```
recursive type(image) function, pointer images::search_img (
            class(image_avl), intent(in) this,
            type(image), intent(in), pointer temp,
            integer, intent(in) img_ig )
```

Method that searches for an image in the tree.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *img↩* | The id of the image to search |
| *_ig* | |
| *temp* | The current node |

**Returns**

The image with the id img_ig

Definition at line 195 of file images.f90.

#### 4.4.1.18 srl()

```
type(image) function, pointer images::srl (
              class(image_avl), intent(in) this,
              type(image), intent(in), pointer t1 )
```

Method that performs a single right rotation.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *t1* | The node to rotate |

**Returns**

The new subtree

Definition at line 216 of file images.f90.

#### 4.4.1.19 srr()

```
type(image) function, pointer images::srr (
              class(image_avl), intent(in) this,
              type(image), intent(in), pointer t1 )
```

Method that performs a single left rotation.

**Parameters**

| | |
|---|---|
| *this* | The image_avl object |
| *t1* | The node to rotate |

**Returns**

The new subtree

Definition at line 230 of file images.f90.

## 4.5 layers Module Reference

### Data Types

- type layer

  *Type to represent a pixel matrix*

- type layers_tree

  *Type to represent a binary tree of layers.*
- type node_layer

  *Type to represent a node of a linked list of layers.*
- type queue

  *Type to represent a queue of layers.*

### Functions/Subroutines

- subroutine gen_inorder (this, tmp, limit, unit)

  *Subroutine to generate the matrix of pixels of a layer.*
- subroutine gen_postorder (this, tmp, limit, unit)

  *Subroutine to generate the matrix of pixels of a layer.*
- subroutine gen_preorder (this, tmp, limit, unit)

  *Subroutine to generate the matrix of pixels of a layer.*
- subroutine traverse_limited (this, order, limit, unit)

  *Subroutine to traverse the layers tree in a limited way.*
- subroutine enqueue (this, layer_val)

  *Subroutine to add a layer to the layers tree.*
- type(layer) function, pointer dequeue (this)

  *Subroutine to remove a layer from the queue.*
- logical function is_empty (this)

  *Function to check if the queue is empty.*
- subroutine traverse_matrix (this)

  *Subroutine to traverse the layers tree.*
- subroutine add (this, new_layer)

  *Subroutine to add a layer to the layers tree.*
- subroutine add_recursive (this, new_layer, tmp)

  *Subroutine to add a layer to the layers tree recursively.*
- subroutine add_copied_val (this, new_layer)

  *Subroutine to add a layer to the layers tree.*
- type(layer) function, pointer search (this, id_searched)

  *Function to search a layer in the layers tree.*
- subroutine preorder (this, tmp)

  *Subroutine to traverse the layers tree in preorder.*
- subroutine inorder (this, tmp)

  *Subroutine to traverse the layers tree in inorder.*
- subroutine postorder (this, tmp)

  *Subroutine to traverse the layers tree in postorder.*
- subroutine gen_dot (this, tmp, unit)

  *Subroutine to generate the dot file of the layers tree.*
- subroutine gen_dot_recursive (this, tmp, unit)

  *Subroutine to generate the dot file of the layers tree recursively.*

- subroutine max_depth (this)

  *Subroutine to calculate the max depth of the layers tree.*
- recursive integer function max_depth_rec (this, root)

  *Function to calculate the max depth of the layers tree recursively.*
- subroutine leaf_layers (this)

  *Subroutine to traverse the leaf layers of the layers tree.*
- subroutine leaf_layers_rec (this, tmp)

  *Subroutine to traverse the leaf layers of the layers tree recursively.*
- subroutine list_layers (this, option)

  *Subroutine to list the layers of the layers tree.*
- subroutine inorder_print (this, tmp)

  *Subroutine to print the layers of the layers tree in inorder.*

## 4.5.1 Function/Subroutine Documentation

### 4.5.1.1 add()

```
subroutine layers::add (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer new_layer )
```

Subroutine to add a layer to the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *new_layer* | Layer to add |

Definition at line 229 of file layers.f90.

### 4.5.1.2 add_copied_val()

```
subroutine layers::add_copied_val (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer new_layer )
```

Subroutine to add a layer to the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *new_layer* | Layer to add |

Definition at line 264 of file layers.f90.

#### 4.5.1.3 add_recursive()

```
subroutine layers::add_recursive (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer new_layer,
             type(layer), intent(inout) tmp )
```

Subroutine to add a layer to the layers tree recursively.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *new_layer* | Layer to add |
| *tmp* | Temporary layer |

Definition at line 243 of file layers.f90.

#### 4.5.1.4 dequeue()

```
type(layer) function, pointer layers::dequeue (
             class(queue), intent(inout) this )
```

Subroutine to remove a layer from the queue.

**Parameters**

| | |
|---|---|
| *this* | Queue |
| *layer_val* | Layer removed |

Definition at line 180 of file layers.f90.

#### 4.5.1.5 enqueue()

```
subroutine layers::enqueue (
             class(queue), intent(inout) this,
             type(layer), intent(in), pointer layer_val )
```

Subroutine to add a layer to the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *new_layer* | Layer to add |

Definition at line 160 of file layers.f90.

#### 4.5.1.6 gen_dot()

```
subroutine layers::gen_dot (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp,
             integer, intent(in) unit )
```

Subroutine to generate the dot file of the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |
| *unit* | Unit to write the dot file |

Definition at line 351 of file layers.f90.

#### 4.5.1.7 gen_dot_recursive()

```
subroutine layers::gen_dot_recursive (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp,
             integer, intent(in) unit )
```

Subroutine to generate the dot file of the layers tree recursively.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |
| *unit* | Unit to write the dot file |

Definition at line 363 of file layers.f90.

#### 4.5.1.8 gen_inorder()

```
subroutine layers::gen_inorder (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp,
             integer, intent(inout) limit,
             integer, intent(in) unit )
```

Subroutine to generate the matrix of pixels of a layer.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |
| *limit* | Limit of layers to traverse |
| *unit* | Unit to write the matrix |

Definition at line 65 of file layers.f90.

#### 4.5.1.9 gen_postorder()

```
subroutine layers::gen_postorder (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp,
             integer, intent(inout) limit,
             integer, intent(in) unit )
```

Subroutine to generate the matrix of pixels of a layer.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |
| *limit* | Limit of layers to traverse |
| *unit* | Unit to write the matrix |

Definition at line 91 of file layers.f90.

#### 4.5.1.10 gen_preorder()

```
subroutine layers::gen_preorder (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp,
             integer, intent(inout) limit,
             integer, intent(in) unit )
```

Subroutine to generate the matrix of pixels of a layer.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |
| *limit* | Limit of layers to traverse |
| *unit* | Unit to write the matrix |

Definition at line 117 of file layers.f90.

#### 4.5.1.11 inorder()

```
subroutine layers::inorder (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the layers tree in inorder.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |

Definition at line 323 of file layers.f90.

#### 4.5.1.12 inorder_print()

```
subroutine layers::inorder_print (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp )
```

Subroutine to print the layers of the layers tree in inorder.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |

Definition at line 460 of file layers.f90.

#### 4.5.1.13 is_empty()

```
logical function layers::is_empty (
             class(queue), intent(in) this )
```

Function to check if the queue is empty.

**Parameters**

| | |
|---|---|
| *this* | Queue |

**Returns**

res Logical value

Logical value

Definition at line 195 of file layers.f90.

#### 4.5.1.14 leaf_layers()

```
subroutine layers::leaf_layers (
            class(layers_tree), intent(inout) this )
```

Subroutine to traverse the leaf layers of the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |

Definition at line 416 of file layers.f90.

#### 4.5.1.15 leaf_layers_rec()

```
subroutine layers::leaf_layers_rec (
            class(layers_tree), intent(inout) this,
            type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the leaf layers of the layers tree recursively.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |

Definition at line 428 of file layers.f90.

#### 4.5.1.16 list_layers()

```
subroutine layers::list_layers (
            class(layers_tree), intent(inout) this,
            integer, intent(in) option )
```

Subroutine to list the layers of the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *option* | Option to list the layers |

Definition at line 443 of file layers.f90.

#### 4.5.1.17 max_depth()

```
subroutine layers::max_depth (
            class(layers_tree), intent(inout) this )
```

Subroutine to calculate the max depth of the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |

Definition at line 382 of file layers.f90.

#### 4.5.1.18 max_depth_rec()

```
recursive integer function layers::max_depth_rec (
            class(layers_tree), intent(inout) this,
            type(layer), intent(in), pointer root )
```

Function to calculate the max depth of the layers tree recursively.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *root* | Root of the tree |

**Returns**

depth Depth of the tree

Definition at line 397 of file layers.f90.

#### 4.5.1.19 postorder()

```
subroutine layers::postorder (
            class(layers_tree), intent(inout) this,
            type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the layers tree in postorder.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |

Definition at line 336 of file layers.f90.

#### 4.5.1.20 preorder()

```
subroutine layers::preorder (
             class(layers_tree), intent(inout) this,
             type(layer), intent(in), pointer tmp )
```

Subroutine to traverse the layers tree in preorder.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *tmp* | Layer |

Definition at line 309 of file layers.f90.

#### 4.5.1.21 search()

```
type(layer) function, pointer layers::search (
             class(layers_tree), intent(inout) this,
             integer, intent(in) id_searched )
```

Function to search a layer in the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *id_searched* | Id of the layer to search |

**Returns**

tmp Layer found

Definition at line 286 of file layers.f90.

#### 4.5.1.22 traverse_limited()

```
subroutine layers::traverse_limited (
             class(layers_tree), intent(inout) this,
             integer, intent(in) order,
             integer, intent(inout) limit,
             integer, intent(in) unit )
```

Subroutine to traverse the layers tree in a limited way.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |
| *order* | Order of traversal |
| *limit* | Limit of layers to traverse |
| *unit* | Unit to write the matrix |

Definition at line 142 of file layers.f90.

#### 4.5.1.23 traverse_matrix()

```
subroutine layers::traverse_matrix (
             class(layers_tree), intent(inout) this )
```

Subroutine to traverse the layers tree.

**Parameters**

| | |
|---|---|
| *this* | Layers tree |

Definition at line 202 of file layers.f90.

## 4.6 pixels Module Reference

### Data Types

- type pixel

  *Type to represent a pixel.*
- type pixel_matrix

  *Type to represent a matrix of pixels.*

### Functions/Subroutines

- subroutine insert (this, x, y, value, color)

  *Insert a pixel in the matrix.*

- type(pixel) function, pointer search_row (this, y)

  *Search a row in the matrix.*
- type(pixel) function, pointer search_column (this, x)

  *Search a column in the matrix.*
- logical function node_exists (this, new_node)

  *Check if a node exists in the matrix.*
- type(pixel) function, pointer insert_row_header (this, y)

  *Insert a row header in the matrix.*
- subroutine insert_in_row (this, new_node, row_header)

  *Insert a pixel in a row.*
- type(pixel) function, pointer insert_column_header (this, x)

  *Insert a column header in the matrix.*
- subroutine insert_in_column (this, new_node, column_header)

  *Insert a pixel in a column.*
- subroutine print_headers (this)

  *Print the headers of the matrix.*
- logical function get_value (this, x, y)

  *Get the value of a pixel.*
- type(pixel) function, pointer get_node (this, x, y)

  *Get the node in the matrix.*
- subroutine self_print (this)

  *Print the matrix.*
- subroutine graph_pixels (this, unit)

  *Generate a graph of the pixels.*
- subroutine gen_matrix (this, g_matrix)

  *Generate a matrix of pixels.*
- subroutine global_m_dot (this, unit)

  *Generate a graph of the pixels.*

## Variables

- integer id = 0

  *Unique identifier for each pixel.*

### 4.6.1 Function/Subroutine Documentation

#### 4.6.1.1 gen_matrix()

```
subroutine pixels::gen_matrix (
             class(pixel_matrix), intent(inout) this,
             type(pixel_matrix), intent(inout) g_matrix )
```

Generate a matrix of pixels.

**Parameters**

| | |
|---|---|
| *this* | The matrix to be converted |
| *g_matrix* | The matrix to be generated |

Definition at line 385 of file pixels.f90.

#### 4.6.1.2 get_node()

```
type(pixel) function, pointer pixels::get_node (
             class(pixel_matrix), intent(inout) this,
             integer, intent(in) x,
             integer, intent(in) y )
```

Get the node in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the node is located |
| *x* | The x coordinate of the node |
| *y* | The y coordinate of the node |

**Returns**

The pointer to the node

Definition at line 257 of file pixels.f90.

#### 4.6.1.3 get_value()

```
logical function pixels::get_value (
             class(pixel_matrix), intent(inout) this,
             integer, intent(in) x,
             integer, intent(in) y )
```

Get the value of a pixel.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the pixel is located |
| *x* | The x coordinate of the pixel |
| *y* | The y coordinate of the pixel |

**Returns**

The value of the pixel

The value of the pixel

Definition at line 230 of file pixels.f90.

#### 4.6.1.4 global_m_dot()

```
subroutine pixels::global_m_dot (
             class(pixel_matrix), intent(inout) this,
             integer, intent(in) unit )
```

Generate a graph of the pixels.

**Parameters**

| | |
|---|---|
| *this* | The matrix to be graphed |
| *unit* | The unit where the graph will be written |

Definition at line 409 of file pixels.f90.

#### 4.6.1.5 graph_pixels()

```
subroutine pixels::graph_pixels (
             class(pixel_matrix), intent(inout) this,
             integer, intent(in) unit )
```

Generate a graph of the pixels.

**Parameters**

| | |
|---|---|
| *this* | The matrix to be graphed |
| *unit* | The unit where the graph will be written |

Definition at line 303 of file pixels.f90.

#### 4.6.1.6 insert()

```
subroutine pixels::insert (
             class(pixel_matrix), intent(inout) this,
             integer, intent(in) x,
             integer, intent(in) y,
             logical, intent(in) value,
             character(len=7), intent(in) color )
```

Insert a pixel in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the pixel will be inserted |
| *x* | The x coordinate of the pixel |
| *y* | The y coordinate of the pixel |
| *value* | The value of the pixel |
| *color* | The color of the pixel |

Definition at line 51 of file pixels.f90.

#### 4.6.1.7 insert_column_header()

```
type(pixel) function, pointer pixels::insert_column_header (
             class(pixel_matrix), intent(inout) this,
             integer, intent(in) x )
```

Insert a column header in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the column header will be inserted |
| *x* | The x coordinate of the column header |

**Returns**

The pointer to the column header

Definition at line 181 of file pixels.f90.

#### 4.6.1.8 insert_in_column()

```
subroutine pixels::insert_in_column (
             class(pixel_matrix), intent(inout) this,
             type(pixel), intent(in), pointer new_node,
             type(pixel), intent(in), pointer column_header )
```

Insert a pixel in a column.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the pixel will be inserted |
| *new_node* | The pixel to be inserted |
| *column_header* | The column header where the pixel will be inserted |

Definition at line 195 of file pixels.f90.

#### 4.6.1.9 insert_in_row()

```
subroutine pixels::insert_in_row (
             class(pixel_matrix), intent(inout) this,
```

```
            type(pixel), intent(in), pointer new_node,
            type(pixel), intent(in), pointer row_header )
```

Insert a pixel in a row.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the pixel will be inserted |
| *new_node* | The pixel to be inserted |
| *row_header* | The row header where the pixel will be inserted |

Definition at line 156 of file pixels.f90.

#### 4.6.1.10 insert_row_header()

```
type(pixel) function, pointer pixels::insert_row_header (
            class(pixel_matrix), intent(inout) this,
            integer, intent(in) y )
```

Insert a row header in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the row header will be inserted |
| *y* | The y coordinate of the row header |

**Returns**

The pointer to the row header

Definition at line 142 of file pixels.f90.

#### 4.6.1.11 node_exists()

```
logical function pixels::node_exists (
            class(pixel_matrix), intent(inout) this,
            type(pixel), intent(in), pointer new_node )
```

Check if a node exists in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the node will be searched |
| *new_node* | The node to be searched |

**Returns**

True if the node exists, False otherwise

Definition at line 116 of file pixels.f90.

#### 4.6.1.12 print_headers()

```
subroutine pixels::print_headers (
            class(pixel_matrix), intent(inout) this )
```

Print the headers of the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the headers will be printed |

Definition at line 218 of file pixels.f90.

#### 4.6.1.13 search_column()

```
type(pixel) function, pointer pixels::search_column (
            class(pixel_matrix), intent(inout) this,
            integer, intent(in) x )
```

Search a column in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the column will be searched |
| *x* | The x coordinate of the column |

**Returns**

The pointer to the column header

Definition at line 102 of file pixels.f90.

#### 4.6.1.14 search_row()

```
type(pixel) function, pointer pixels::search_row (
            class(pixel_matrix), intent(inout) this,
            integer, intent(in) y )
```

Search a row in the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix where the row will be searched |
| *y* | The y coordinate of the row |

**Returns**

The pointer to the row header

Definition at line 88 of file pixels.f90.

#### 4.6.1.15 self_print()

```
subroutine pixels::self_print (
             class(pixel_matrix), intent(inout) this )
```

Print the matrix.

**Parameters**

| | |
|---|---|
| *this* | The matrix to be printed |

Definition at line 281 of file pixels.f90.

### 4.6.2 Variable Documentation

#### 4.6.2.1 id

```
integer pixels::id = 0
```

Unique identifier for each pixel.

Definition at line 10 of file pixels.f90.

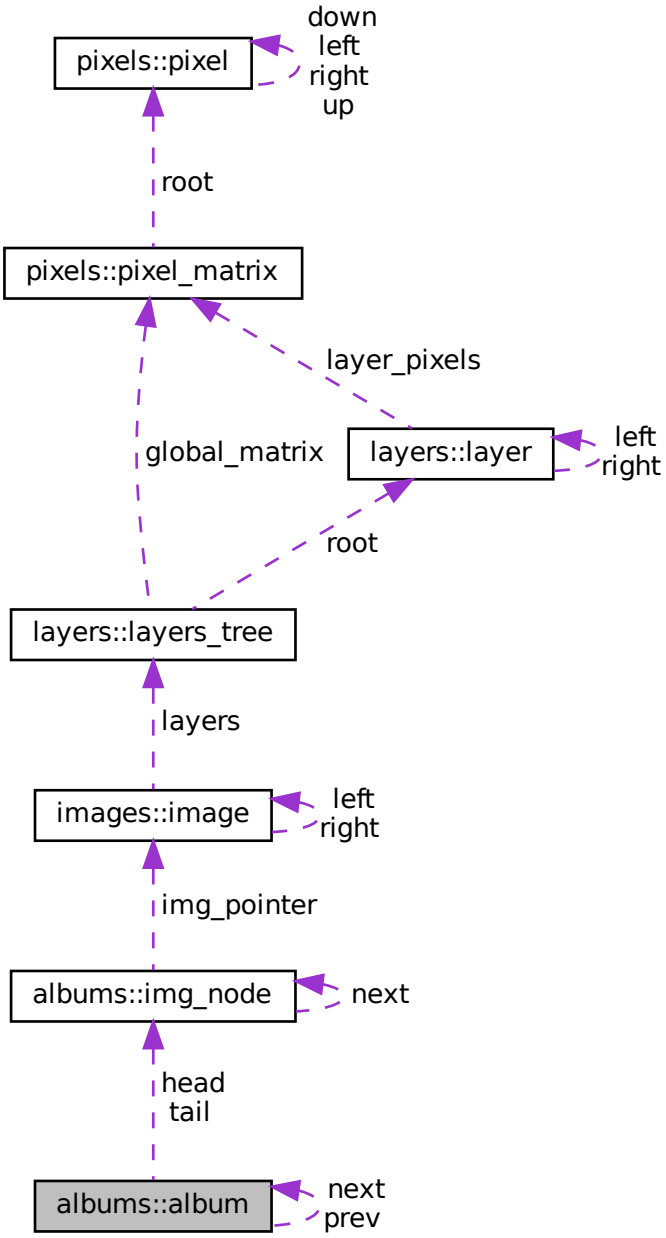
# Chapter 5

# Data Type Documentation

## 5.1 albums::album Type Reference

Type to store the album.

Collaboration diagram for albums::album:




## Public Member Functions

- procedure add_image
- procedure remove_image
- procedure get_image
- procedure show_images

## Public Attributes

- integer id
  *Album id.*
- character(:), allocatable name
  *Album name.*
- type(album), pointer next => null()
  *Pointer to the next album.*
- type(album), pointer prev => null()
  *Pointer to the previous album.*
- type(img_node), pointer head => null()
  *Pointer to the first image.*
- type(img_node), pointer tail => null()
  *Pointer to the last image.*
- integer size = 0
  *Number of images in the album.*

### 5.1.1 Detailed Description

Type to store the album.

Definition at line 18 of file albums.f90.

### 5.1.2 Member Function/Subroutine Documentation

#### 5.1.2.1 add_image()

```
procedure albums::album::add_image
```

Definition at line 27 of file albums.f90.

#### 5.1.2.2 get_image()

```
procedure albums::album::get_image
```

Definition at line 29 of file albums.f90.

#### 5.1.2.3 remove_image()

```
procedure albums::album::remove_image
```

Definition at line 28 of file albums.f90.

#### 5.1.2.4 show_images()

```
procedure albums::album::show_images
```

Definition at line 30 of file albums.f90.

### 5.1.3 Member Data Documentation

#### 5.1.3.1 head

```
type(img_node), pointer albums::album::head => null()
```

Pointer to the first image.

Definition at line 23 of file albums.f90.

#### 5.1.3.2 id

```
integer albums::album::id
```

Album id.

Definition at line 19 of file albums.f90.

#### 5.1.3.3 name

```
character(:), allocatable albums::album::name
```

Album name.

Definition at line 20 of file albums.f90.

#### 5.1.3.4 next

```
type(album), pointer albums::album::next => null()
```

Pointer to the next album.

Definition at line 21 of file albums.f90.

#### 5.1.3.5 prev

```
type(album), pointer albums::album::prev => null()
```

Pointer to the previous album.

Definition at line 22 of file albums.f90.

#### 5.1.3.6 size

```
integer albums::album::size = 0
```

Number of images in the album.

Definition at line 25 of file albums.f90.

#### 5.1.3.7 tail

```
type(img_node), pointer albums::album::tail => null()
```

Pointer to the last image.

Definition at line 24 of file albums.f90.

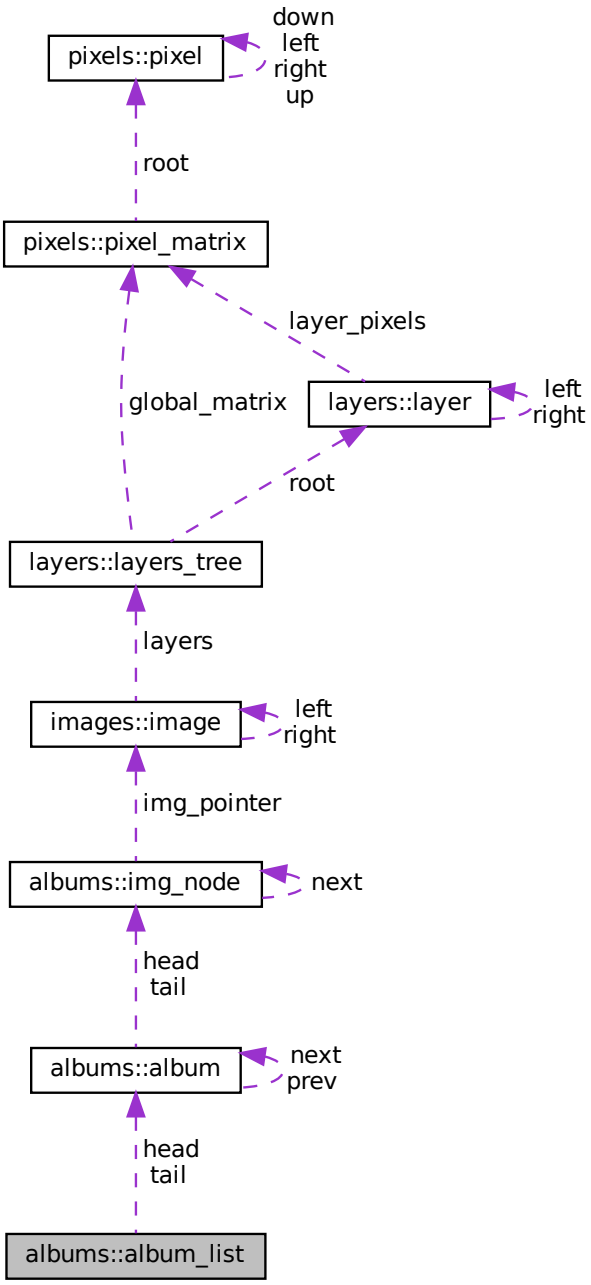
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90

## 5.2 albums::album_list Type Reference

Type to store the album list.

Collaboration diagram for albums::album_list:




## Public Member Functions

- procedure new_album
- procedure remove_album
- procedure get_album
- procedure search_in_album
- procedure add_image_to_album

- procedure remove_image_from_album
- procedure show_albums
- procedure show_album_images
- procedure gen_album_graph

## Public Attributes

- type(album), pointer head => null()

  *Pointer to the first album.*
- type(album), pointer tail => null()

  *Pointer to the last album.*
- integer size = 0

  *Number of albums in the list.*

### 5.2.1 Detailed Description

Type to store the album list.

Definition at line 33 of file albums.f90.

### 5.2.2 Member Function/Subroutine Documentation

#### 5.2.2.1 add_image_to_album()

```
procedure albums::album_list::add_image_to_album
```

Definition at line 42 of file albums.f90.

#### 5.2.2.2 gen_album_graph()

```
procedure albums::album_list::gen_album_graph
```

Definition at line 46 of file albums.f90.

#### 5.2.2.3 get_album()

```
procedure albums::album_list::get_album
```

Definition at line 40 of file albums.f90.

#### 5.2.2.4 new_album()

```
procedure albums::album_list::new_album
```

Definition at line 38 of file albums.f90.

#### 5.2.2.5 remove_album()

```
procedure albums::album_list::remove_album
```

Definition at line 39 of file albums.f90.

#### 5.2.2.6 remove_image_from_album()

```
procedure albums::album_list::remove_image_from_album
```

Definition at line 43 of file albums.f90.

#### 5.2.2.7 search_in_album()

```
procedure albums::album_list::search_in_album
```

Definition at line 41 of file albums.f90.

#### 5.2.2.8 show_album_images()

```
procedure albums::album_list::show_album_images
```

Definition at line 45 of file albums.f90.

#### 5.2.2.9 show_albums()

```
procedure albums::album_list::show_albums
```

Definition at line 44 of file albums.f90.

### 5.2.3 Member Data Documentation

#### 5.2.3.1 head

```
type(album), pointer albums::album_list::head => null()
```

Pointer to the first album.

Definition at line 34 of file albums.f90.

#### 5.2.3.2 size

```
integer albums::album_list::size = 0
```

Number of albums in the list.

Definition at line 36 of file albums.f90.

#### 5.2.3.3 tail

```
type(album), pointer albums::album_list::tail => null()
```

Pointer to the last album.

Definition at line 35 of file albums.f90.

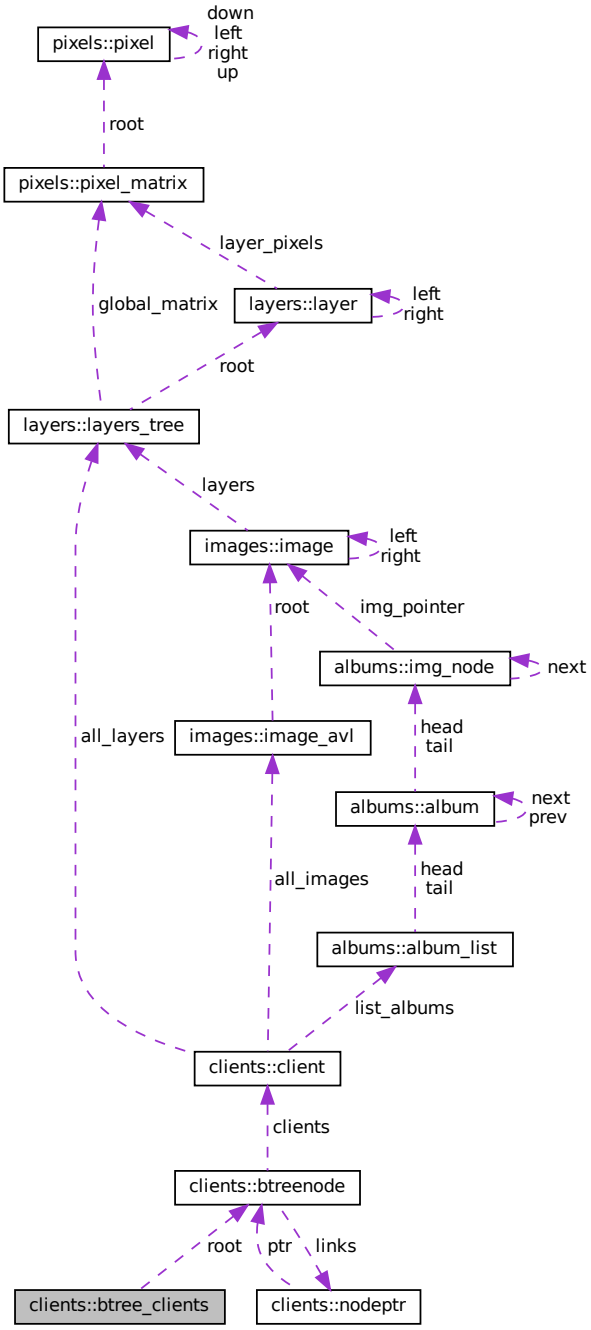
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90

## 5.3 clients::btree_clients Type Reference

Type to represent a B-tree.

Collaboration diagram for clients::btree_clients:




### Public Member Functions

- procedure add_client

- procedure search_client
- procedure delete_client
- procedure delete_client_rec
- procedure create_node
- procedure traversal
- procedure clients_dot
- procedure amplitude_traversal

## Public Attributes

- type(btreenode), pointer root => null()

  *Pointer to the root of the tree.*

### 5.3.1 Detailed Description

Type to represent a B-tree.

Definition at line 37 of file clients.f90.

### 5.3.2 Member Function/Subroutine Documentation

#### 5.3.2.1 add_client()

```
procedure clients::btree_clients::add_client
```

Definition at line 40 of file clients.f90.

#### 5.3.2.2 amplitude_traversal()

```
procedure clients::btree_clients::amplitude_traversal
```

Definition at line 47 of file clients.f90.

#### 5.3.2.3 clients_dot()

```
procedure clients::btree_clients::clients_dot
```

Definition at line 46 of file clients.f90.

#### 5.3.2.4 create_node()

```
procedure clients::btree_clients::create_node
```

Definition at line 44 of file clients.f90.

#### 5.3.2.5 delete_client()

```
procedure clients::btree_clients::delete_client
```

Definition at line 42 of file clients.f90.

#### 5.3.2.6 delete_client_rec()

```
procedure clients::btree_clients::delete_client_rec
```

Definition at line 43 of file clients.f90.

#### 5.3.2.7 search_client()

```
procedure clients::btree_clients::search_client
```

Definition at line 41 of file clients.f90.

#### 5.3.2.8 traversal()

```
procedure clients::btree_clients::traversal
```

Definition at line 45 of file clients.f90.

### 5.3.3 Member Data Documentation

#### 5.3.3.1 root

```
type(btreenode), pointer clients::btree_clients::root => null()
```

Pointer to the root of the tree.

Definition at line 38 of file clients.f90.

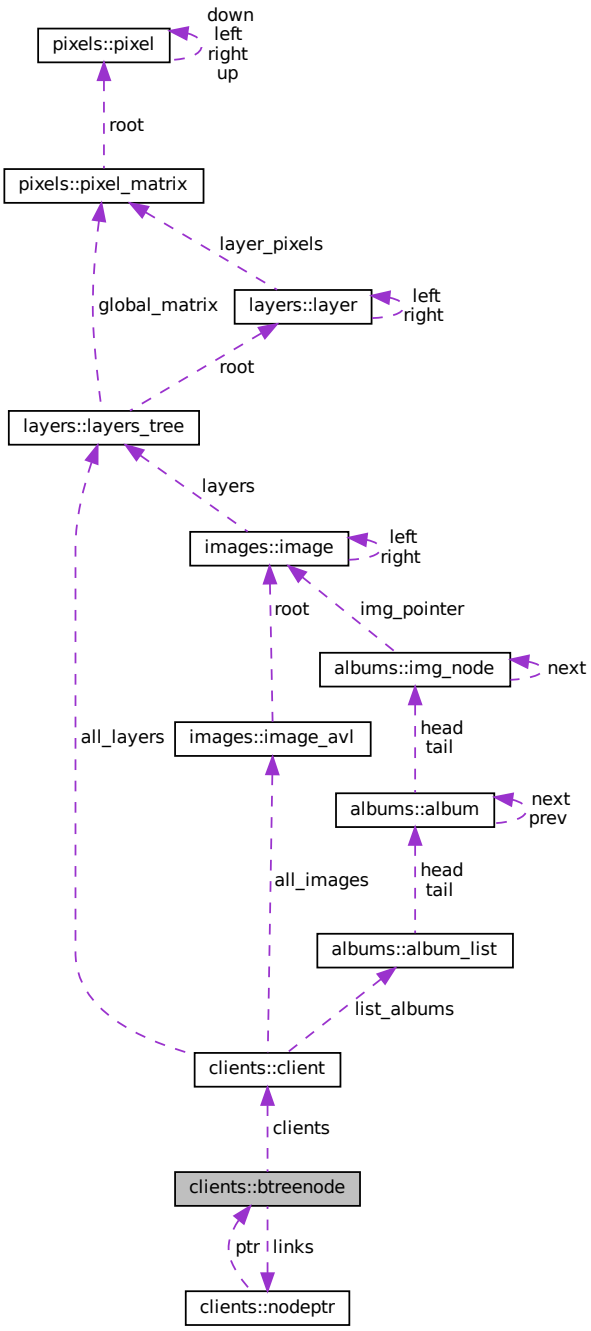
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90

## 5.4 clients::btreenode Type Reference

Type to represent a B-tree node.

Collaboration diagram for clients::btreenode:




## Public Member Functions

- procedure find

## Public Attributes

- integer id

*Id of the node.*

- type(client), dimension(0:5) clients

  *Array to store the clients of the node.*

- integer num = 0

  *Number of clients in the node.*

- type(nodeptr), dimension(0:5) links

  *Array to store the links to other nodes.*

### 5.4.1 Detailed Description

Type to represent a B-tree node.

Definition at line 28 of file clients.f90.

### 5.4.2 Member Function/Subroutine Documentation

#### 5.4.2.1 find()

```
procedure clients::btreenode::find
```

Definition at line 34 of file clients.f90.

### 5.4.3 Member Data Documentation

#### 5.4.3.1 clients

```
type(client), dimension(0:5) clients::btreenode::clients
```

Array to store the clients of the node.

Definition at line 30 of file clients.f90.

#### 5.4.3.2 id

```
integer clients::btreenode::id
```

Id of the node.

Definition at line 29 of file clients.f90.

#### 5.4.3.3 links

```
type(nodeptr), dimension(0:5) clients::btreenode::links
```

Array to store the links to other nodes.

Definition at line 32 of file clients.f90.

#### 5.4.3.4 num

```
integer clients::btreenode::num = 0
```

Number of clients in the node.

Definition at line 31 of file clients.f90.

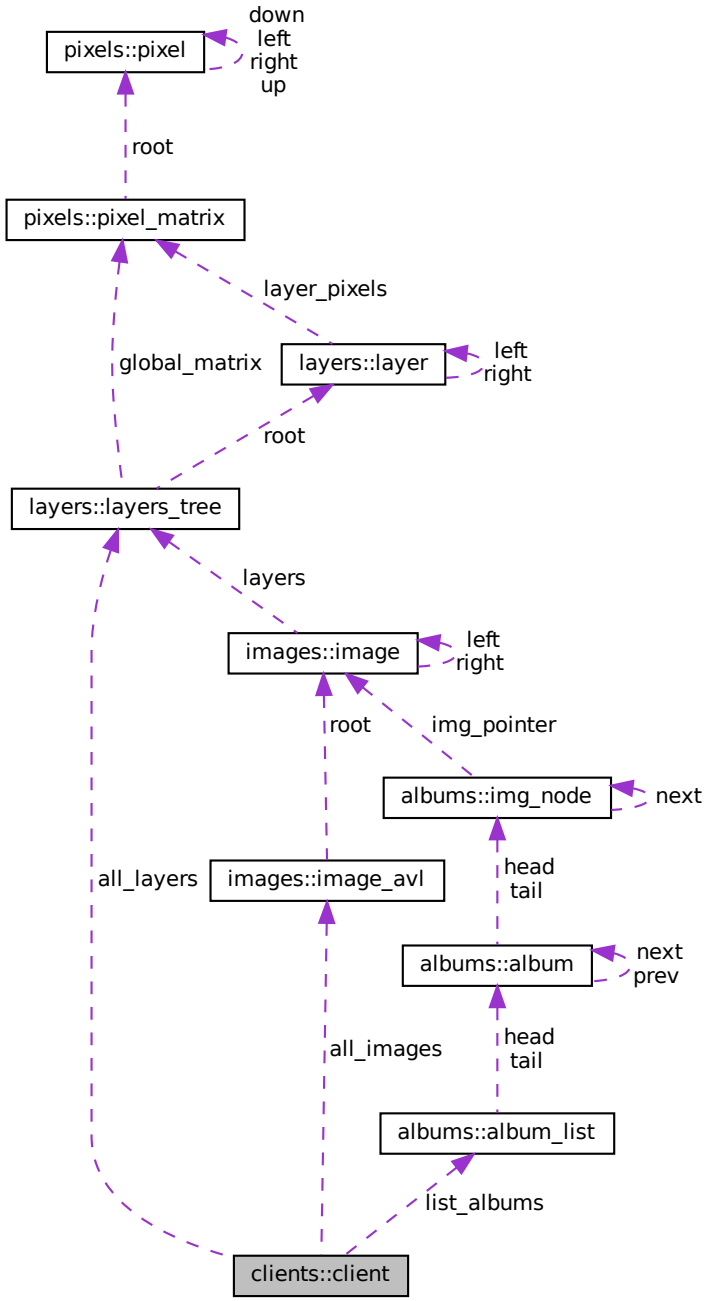
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90

## 5.5 clients::client Type Reference

Type to represent a client.

Collaboration diagram for clients::client:




## Public Attributes

- character(:), allocatable name

  *Name of the client.*
- integer(kind=8) dpi

  *DPI of the client.*
- character(:), allocatable password

*Password of the client.*

- type(image_avl) all_images

  *AVL tree to store the images of the client.*

- type(album_list) list_albums

  *List to store the albums of the client.*

- type(layers_tree) all_layers

  *Tree to store the layers of the client.*

### 5.5.1 Detailed Description

Type to represent a client.

Definition at line 15 of file clients.f90.

### 5.5.2 Member Data Documentation

#### 5.5.2.1 all_images

```
type(image_avl) clients::client::all_images
```

AVL tree to store the images of the client.

Definition at line 19 of file clients.f90.

#### 5.5.2.2 all_layers

```
type(layers_tree) clients::client::all_layers
```

Tree to store the layers of the client.

Definition at line 21 of file clients.f90.

#### 5.5.2.3 dpi

```
integer(kind=8) clients::client::dpi
```

DPI of the client.

Definition at line 17 of file clients.f90.

#### 5.5.2.4 list_albums

```
type(album_list) clients::client::list_albums
```

List to store the albums of the client.

Definition at line 20 of file clients.f90.

#### 5.5.2.5 name

```
character(:), allocatable clients::client::name
```

Name of the client.

Definition at line 16 of file clients.f90.

#### 5.5.2.6 password

```
character(:), allocatable clients::client::password
```

Password of the client.

Definition at line 18 of file clients.f90.

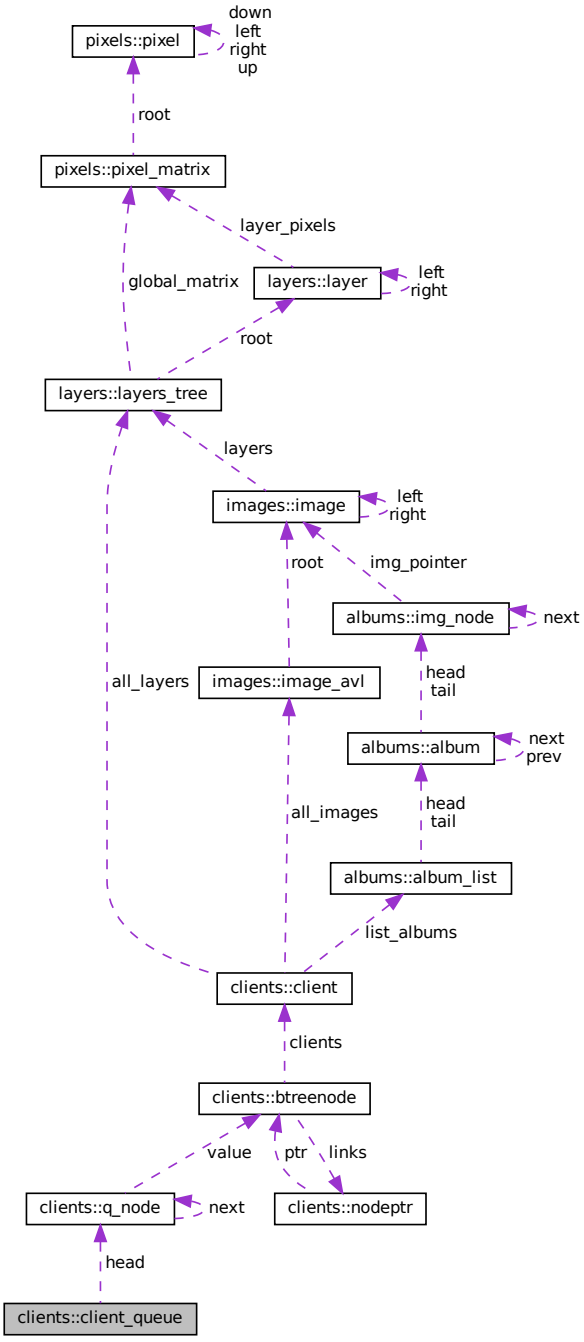
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90

## 5.6 clients::client_queue Type Reference

Type to represent a client queue.

Collaboration diagram for clients::client_queue:




## Public Member Functions

- procedure enqueue
- procedure dequeue
- procedure is_empty

## Public Attributes

- type(q_node), pointer head => null()
  *Pointer to the head of the queue.*

### 5.6.1 Detailed Description

Type to represent a client queue.

Definition at line 55 of file clients.f90.

### 5.6.2 Member Function/Subroutine Documentation

#### 5.6.2.1 dequeue()

```
procedure clients::client_queue::dequeue
```

Definition at line 59 of file clients.f90.

#### 5.6.2.2 enqueue()

```
procedure clients::client_queue::enqueue
```

Definition at line 58 of file clients.f90.

#### 5.6.2.3 is_empty()

```
procedure clients::client_queue::is_empty
```

Definition at line 60 of file clients.f90.

### 5.6.3 Member Data Documentation

#### 5.6.3.1 head

```
type(q_node), pointer clients::client_queue::head => null()
```

Pointer to the head of the queue.

Definition at line 56 of file clients.f90.

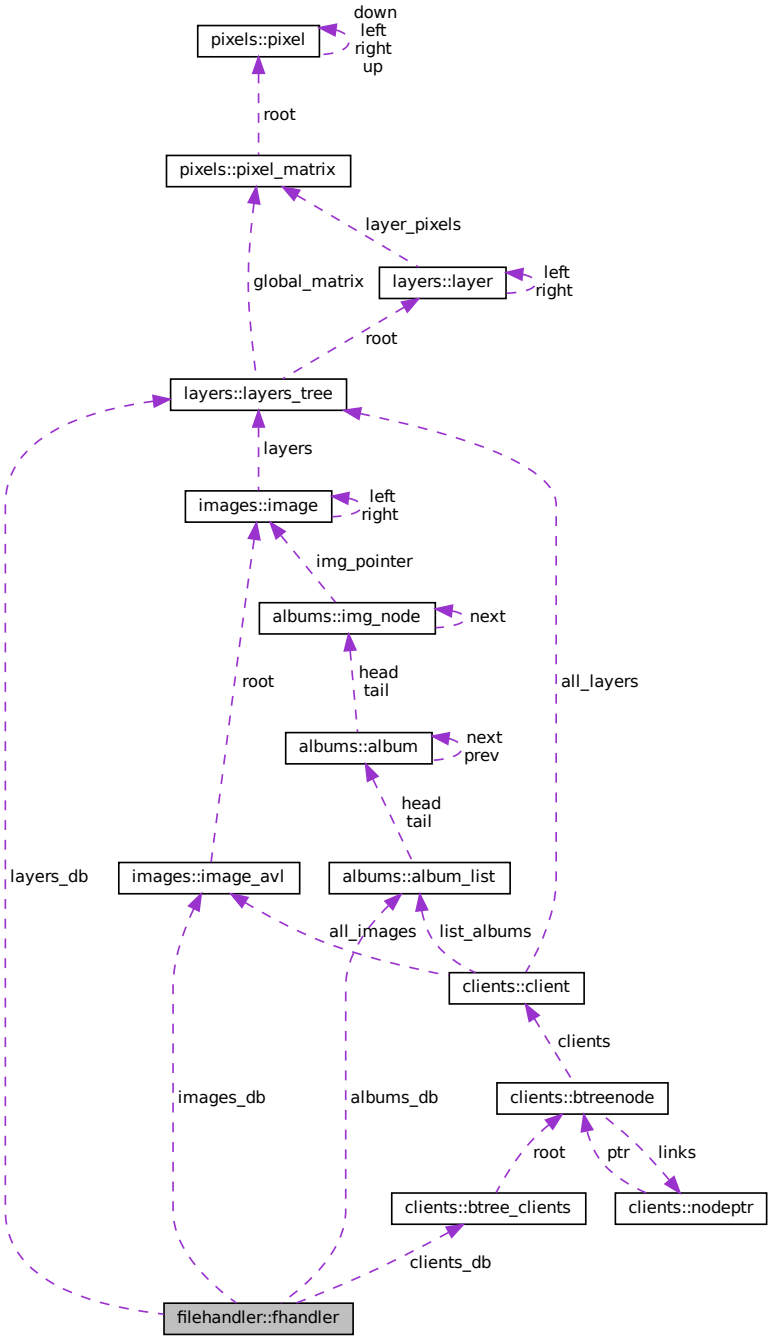
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90

## 5.7 filehandler::fhandler Type Reference

Type that handles the reading of the json files and the initialization of the data structures.

Collaboration diagram for filehandler::fhandler:




## Public Member Functions

- procedure initialize_admin
- procedure initialize_user
- procedure set_user
- procedure read_imgs
- procedure read_layers

- procedure read_albums
- procedure read_clients

## Public Attributes

- type(album_list), pointer albums_db => null()
  *Pointer to the albums database.*
- type(image_avl), pointer images_db => null()
  *Pointer to the images database.*
- type(btree_clients), pointer clients_db => null()
  *Pointer to the clients database.*
- type(layers_tree), pointer layers_db => null()
  *Pointer to the layers database.*

### 5.7.1 Detailed Description

Type that handles the reading of the json files and the initialization of the data structures.

Definition at line 17 of file filehandler.f90.

### 5.7.2 Member Function/Subroutine Documentation

#### 5.7.2.1 initialize_admin()

```
procedure filehandler::fhandler::initialize_admin
```

Definition at line 23 of file filehandler.f90.

#### 5.7.2.2 initialize_user()

```
procedure filehandler::fhandler::initialize_user
```

Definition at line 24 of file filehandler.f90.

#### 5.7.2.3 read_albums()

```
procedure filehandler::fhandler::read_albums
```

Definition at line 28 of file filehandler.f90.

#### 5.7.2.4 read_clients()

```
procedure filehandler::fhandler::read_clients
```

Definition at line 29 of file filehandler.f90.

#### 5.7.2.5 read_imgs()

```
procedure filehandler::fhandler::read_imgs
```

Definition at line 26 of file filehandler.f90.

#### 5.7.2.6 read_layers()

```
procedure filehandler::fhandler::read_layers
```

Definition at line 27 of file filehandler.f90.

#### 5.7.2.7 set_user()

```
procedure filehandler::fhandler::set_user
```

Definition at line 25 of file filehandler.f90.

### 5.7.3 Member Data Documentation

#### 5.7.3.1 albums_db

```
type(album_list), pointer filehandler::fhandler::albums_db => null()
```

Pointer to the albums database.

Definition at line 18 of file filehandler.f90.

#### 5.7.3.2 clients_db

```
type(btree_clients), pointer filehandler::fhandler::clients_db => null()
```

Pointer to the clients database.

Definition at line 20 of file filehandler.f90.

#### 5.7.3.3 images_db

```
type(image_avl), pointer filehandler::fhandler::images_db => null()
```

Pointer to the images database.

Definition at line 19 of file filehandler.f90.

#### 5.7.3.4 layers_db

```
type(layers_tree), pointer filehandler::fhandler::layers_db => null()
```

Pointer to the layers database.

Definition at line 21 of file filehandler.f90.

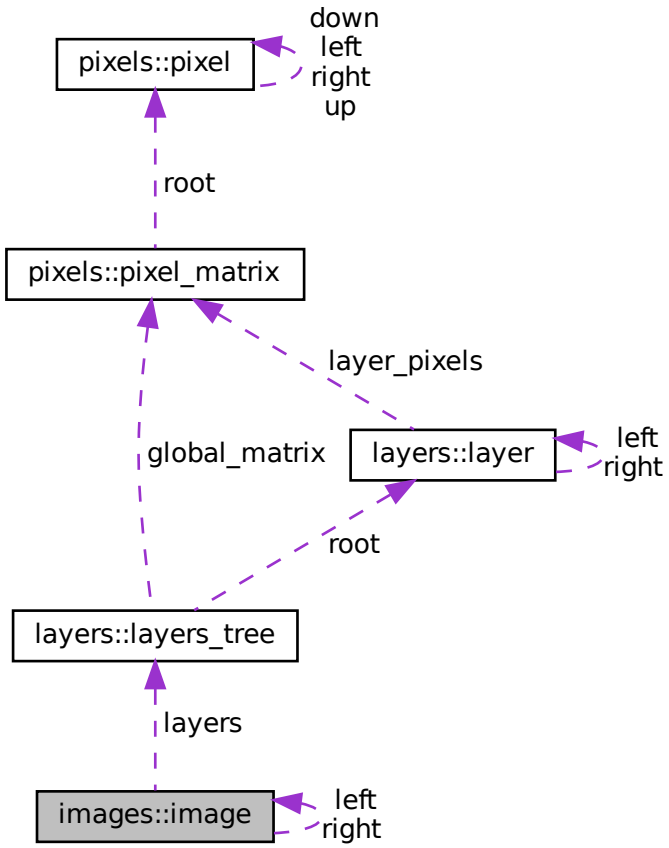
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/filehandler.f90

## 5.8 images::image Type Reference

Type that represents a node in the AVL tree.

Collaboration diagram for images::image:




## Public Member Functions

- procedure add_layer

## Public Attributes

- integer id

  *The id of the image.*
- integer height

  *The height of the node.*
- integer layers_count = 0

  *The number of layers in the image.*
- type(layers_tree) layers

  *The layers of the image.*
- type(image), pointer left => null()

  *The left child of the node.*
- type(image), pointer right => null()

  *The right child of the nodes.*

### 5.8.1 Detailed Description

Type that represents a node in the AVL tree.

Definition at line 11 of file images.f90.

### 5.8.2 Member Function/Subroutine Documentation

#### 5.8.2.1 add_layer()

```
procedure images::image::add_layer
```

Definition at line 19 of file images.f90.

### 5.8.3 Member Data Documentation

#### 5.8.3.1 height

```
integer images::image::height
```

The height of the node.

Definition at line 13 of file images.f90.

#### 5.8.3.2 id

```
integer images::image::id
```

The id of the image.

Definition at line 12 of file images.f90.

#### 5.8.3.3 layers

```
type(layers_tree) images::image::layers
```

The layers of the image.

Definition at line 15 of file images.f90.

#### 5.8.3.4 layers_count

```
integer images::image::layers_count = 0
```

The number of layers in the image.

Definition at line 14 of file images.f90.

#### 5.8.3.5 left

```
type(image), pointer images::image::left => null()
```

The left child of the node.

Definition at line 16 of file images.f90.

#### 5.8.3.6 right

```
type(image), pointer images::image::right => null()
```

The right child of the nodes.

Definition at line 17 of file images.f90.

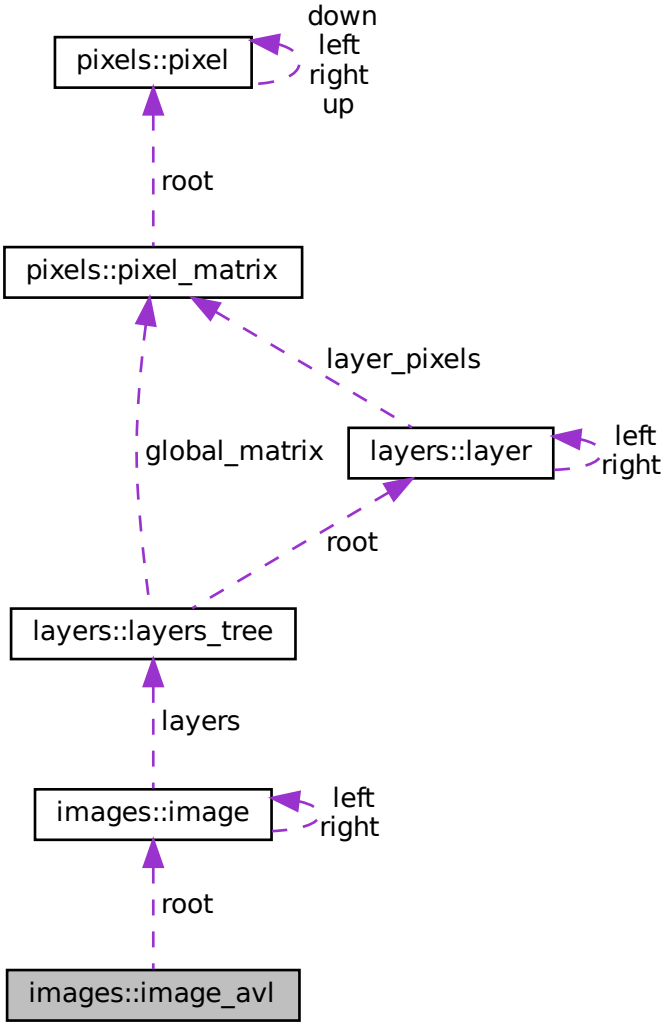
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90

## 5.9 images::image_avl Type Reference

Type that represents an AVL tree of images.

Collaboration diagram for images::image_avl:




## Public Member Functions

- procedure add_img
- procedure add_img_rec
- procedure search_img
- procedure srl
- procedure srr
- procedure drl
- procedure drr
- procedure get_max
- procedure min_child
- procedure get_height
- procedure get_dot
- procedure get_dot_rec

- procedure delete_img
- procedure delete_img_rec
- procedure print_images
- procedure gen_tree_subtree
- procedure gen_img_traversal

## Public Attributes

- type(image), pointer root => null()

  *The root of the tree.*
- integer total = 0

  *The total number of images in the tree.*

### 5.9.1 Detailed Description

Type that represents an AVL tree of images.

Definition at line 22 of file images.f90.

### 5.9.2 Member Function/Subroutine Documentation

#### 5.9.2.1 add_img()

```
procedure images::image_avl::add_img
```

Definition at line 26 of file images.f90.

#### 5.9.2.2 add_img_rec()

```
procedure images::image_avl::add_img_rec
```

Definition at line 27 of file images.f90.

#### 5.9.2.3 delete_img()

```
procedure images::image_avl::delete_img
```

Definition at line 38 of file images.f90.

#### 5.9.2.4 delete_img_rec()

```
procedure images::image_avl::delete_img_rec
```

Definition at line 39 of file images.f90.

#### 5.9.2.5 drl()

```
procedure images::image_avl::drl
```

Definition at line 31 of file images.f90.

#### 5.9.2.6 drr()

```
procedure images::image_avl::drr
```

Definition at line 32 of file images.f90.

#### 5.9.2.7 gen_img_traversal()

```
procedure images::image_avl::gen_img_traversal
```

Definition at line 42 of file images.f90.

#### 5.9.2.8 gen_tree_subtree()

```
procedure images::image_avl::gen_tree_subtree
```

Definition at line 41 of file images.f90.

#### 5.9.2.9 get_dot()

```
procedure images::image_avl::get_dot
```

Definition at line 36 of file images.f90.

#### 5.9.2.10 get_dot_rec()

```
procedure images::image_avl::get_dot_rec
```

Definition at line 37 of file images.f90.

#### 5.9.2.11 get_height()

```
procedure images::image_avl::get_height
```

Definition at line 35 of file images.f90.

#### 5.9.2.12 get_max()

```
procedure images::image_avl::get_max
```

Definition at line 33 of file images.f90.

#### 5.9.2.13 min_child()

```
procedure images::image_avl::min_child
```

Definition at line 34 of file images.f90.

#### 5.9.2.14 print_images()

```
procedure images::image_avl::print_images
```

Definition at line 40 of file images.f90.

#### 5.9.2.15 search_img()

```
procedure images::image_avl::search_img
```

Definition at line 28 of file images.f90.

#### 5.9.2.16 srl()

```
procedure images::image_avl::srl
```

Definition at line 29 of file images.f90.

#### 5.9.2.17 srr()

```
procedure images::image_avl::srr
```

Definition at line 30 of file images.f90.

### 5.9.3 Member Data Documentation

#### 5.9.3.1 root

```
type(image), pointer images::image_avl::root => null()
```

The root of the tree.

Definition at line 23 of file images.f90.

#### 5.9.3.2 total

```
integer images::image_avl::total = 0
```

The total number of images in the tree.

Definition at line 24 of file images.f90.

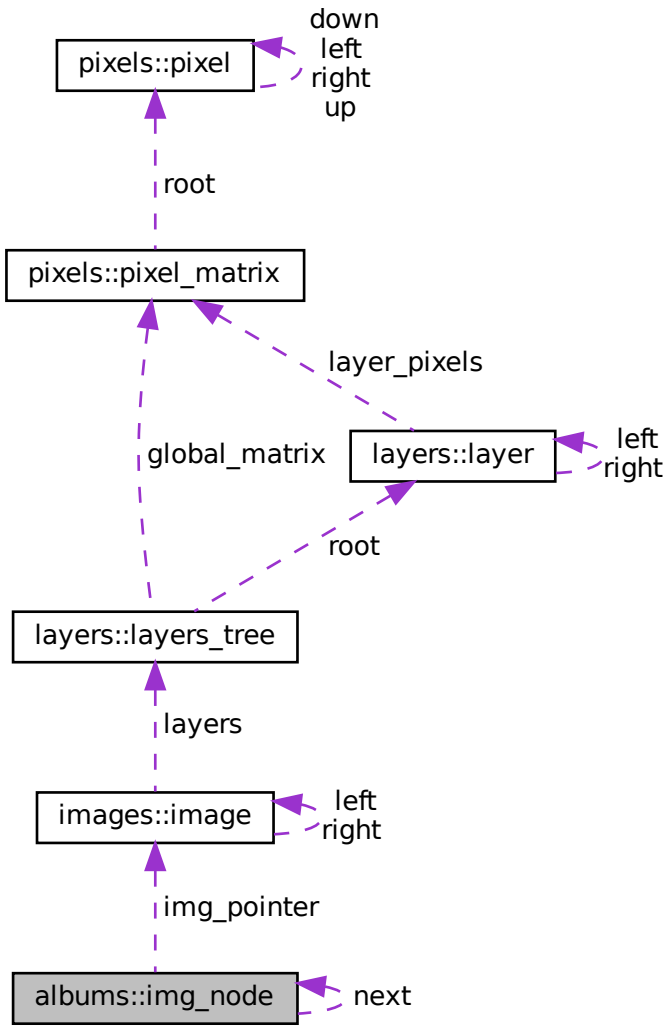
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90

## 5.10 albums::img_node Type Reference

Type to store the image node.

Collaboration diagram for albums::img_node:




### Public Attributes

- integer id

  *Image id.*
- type(image), pointer img_pointer => null()

  *Pointer to the image.*
- type(img_node), pointer next => null()

  *Pointer to the next image.*

### 5.10.1 Detailed Description

Type to store the image node.

Definition at line 12 of file albums.f90.

### 5.10.2 Member Data Documentation

#### 5.10.2.1 id

```
integer albums::img_node::id
```

Image id.

Definition at line 13 of file albums.f90.

#### 5.10.2.2 img_pointer

```
type(image), pointer albums::img_node::img_pointer => null()
```

Pointer to the image.

Definition at line 14 of file albums.f90.

#### 5.10.2.3 next

```
type(img_node), pointer albums::img_node::next => null()
```

Pointer to the next image.

Definition at line 15 of file albums.f90.

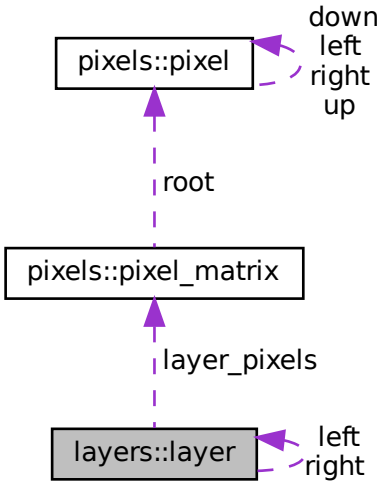
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90

## 5.11 layers::layer Type Reference

Type to represent a pixel matrix

Collaboration diagram for layers::layer:




### Public Attributes

- integer id
  *Layer id.*
- integer pixels_count
  *Number of pixels in the layer.*
- type(pixel_matrix) layer_pixels
  *Matrix of pixels in the layer.*
- type(layer), pointer left => null()
  *Pointer to the left child layer.*
- type(layer), pointer right => null()
  *Pointer to the right child layer.*

### 5.11.1 Detailed Description

Type to represent a pixel matrix

Definition at line 12 of file layers.f90.

### 5.11.2 Member Data Documentation

#### 5.11.2.1 id

```
integer layers::layer::id
```

Layer id.

Definition at line 13 of file layers.f90.

#### 5.11.2.2 layer_pixels

```
type(pixel_matrix) layers::layer::layer_pixels
```

Matrix of pixels in the layer.

Definition at line 15 of file layers.f90.

#### 5.11.2.3 left

```
type(layer), pointer layers::layer::left => null()
```

Pointer to the left child layer.

Definition at line 16 of file layers.f90.

#### 5.11.2.4 pixels_count

```
integer layers::layer::pixels_count
```

Number of pixels in the layer.

Definition at line 14 of file layers.f90.

#### 5.11.2.5 right

```
type(layer), pointer layers::layer::right => null()
```

Pointer to the right child layer.

Definition at line 17 of file layers.f90.

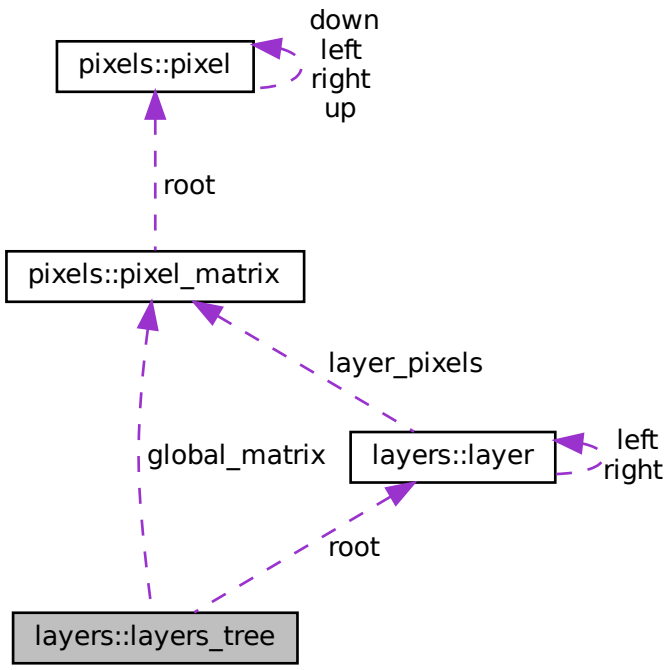
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90

## 5.12 layers::layers_tree Type Reference

Type to represent a binary tree of layers.

Collaboration diagram for layers::layers_tree:



## Public Member Functions

- procedure add
- procedure add_recursive
- procedure add_copied_val
- procedure preorder
- procedure inorder
- procedure postorder
- procedure gen_dot
- procedure gen_dot_recursive
- procedure search
- procedure traverse_matrix
- procedure max_depth
- procedure max_depth_rec
- procedure leaf_layers
- procedure leaf_layers_rec
- procedure list_layers
- procedure inorder_print
- procedure traverse_limited
- procedure gen_preorder
- procedure gen_inorder
- procedure gen_postorder

## Public Attributes

- type(layer), pointer root => null()

  *Pointer to the root layer.*
- type(pixel_matrix) global_matrix

  *Global matrix of pixels.*
- integer total = 0

  *Total number of layers in the tree.*

### 5.12.1 Detailed Description

Type to represent a binary tree of layers.

Definition at line 20 of file layers.f90.

### 5.12.2 Member Function/Subroutine Documentation

#### 5.12.2.1 add()

```
procedure layers::layers_tree::add
```

Definition at line 25 of file layers.f90.

#### 5.12.2.2 add_copied_val()

```
procedure layers::layers_tree::add_copied_val
```

Definition at line 27 of file layers.f90.

#### 5.12.2.3 add_recursive()

```
procedure layers::layers_tree::add_recursive
```

Definition at line 26 of file layers.f90.

#### 5.12.2.4 gen_dot()

```
procedure layers::layers_tree::gen_dot
```

Definition at line 31 of file layers.f90.

#### 5.12.2.5 gen_dot_recursive()

```
procedure layers::layers_tree::gen_dot_recursive
```

Definition at line 32 of file layers.f90.

#### 5.12.2.6 gen_inorder()

```
procedure layers::layers_tree::gen_inorder
```

Definition at line 43 of file layers.f90.

#### 5.12.2.7 gen_postorder()

```
procedure layers::layers_tree::gen_postorder
```

Definition at line 44 of file layers.f90.

#### 5.12.2.8 gen_preorder()

```
procedure layers::layers_tree::gen_preorder
```

Definition at line 42 of file layers.f90.

#### 5.12.2.9 inorder()

```
procedure layers::layers_tree::inorder
```

Definition at line 29 of file layers.f90.

#### 5.12.2.10 inorder_print()

```
procedure layers::layers_tree::inorder_print
```

Definition at line 40 of file layers.f90.

#### 5.12.2.11 leaf_layers()

```
procedure layers::layers_tree::leaf_layers
```

Definition at line 37 of file layers.f90.

#### 5.12.2.12 leaf_layers_rec()

```
procedure layers::layers_tree::leaf_layers_rec
```

Definition at line 38 of file layers.f90.

#### 5.12.2.13 list_layers()

```
procedure layers::layers_tree::list_layers
```

Definition at line 39 of file layers.f90.

#### 5.12.2.14 max_depth()

```
procedure layers::layers_tree::max_depth
```

Definition at line 35 of file layers.f90.

#### 5.12.2.15 max_depth_rec()

```
procedure layers::layers_tree::max_depth_rec
```

Definition at line 36 of file layers.f90.

#### 5.12.2.16 postorder()

```
procedure layers::layers_tree::postorder
```

Definition at line 30 of file layers.f90.

#### 5.12.2.17 preorder()

```
procedure layers::layers_tree::preorder
```

Definition at line 28 of file layers.f90.

#### 5.12.2.18 search()

```
procedure layers::layers_tree::search
```

Definition at line 33 of file layers.f90.

#### 5.12.2.19 traverse_limited()

```
procedure layers::layers_tree::traverse_limited
```

Definition at line 41 of file layers.f90.

#### 5.12.2.20 traverse_matrix()

```
procedure layers::layers_tree::traverse_matrix
```

Definition at line 34 of file layers.f90.

### 5.12.3 Member Data Documentation

#### 5.12.3.1 global_matrix

```
type(pixel_matrix) layers::layers_tree::global_matrix
```

Global matrix of pixels.

Definition at line 22 of file layers.f90.

#### 5.12.3.2 root

```
type(layer), pointer layers::layers_tree::root => null()
```

Pointer to the root layer.

Definition at line 21 of file layers.f90.

#### 5.12.3.3 total

```
integer layers::layers_tree::total = 0
```

Total number of layers in the tree.

Definition at line 23 of file layers.f90.

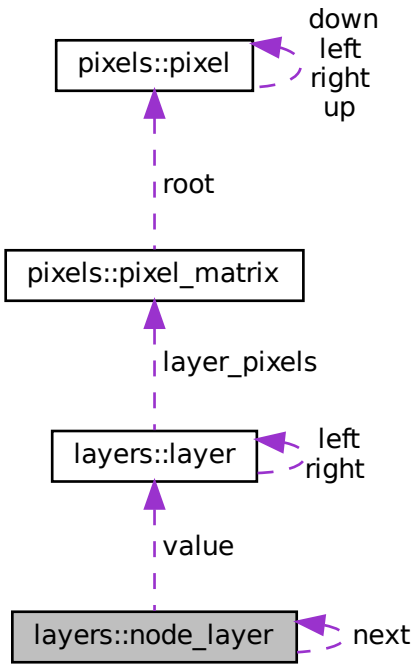
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90

## 5.13 layers::node_layer Type Reference

Type to represent a node of a linked list of layers.

Collaboration diagram for layers::node_layer:




### Public Attributes

- type(layer), pointer value

  *Pointer to a layer.*
- type(node_layer), pointer next => null()

  *Pointer to the next node.*

### 5.13.1 Detailed Description

Type to represent a node of a linked list of layers.

Definition at line 47 of file layers.f90.

### 5.13.2 Member Data Documentation

#### 5.13.2.1 next

```
type(node_layer), pointer layers::node_layer::next => null()
```

Pointer to the next node.

Definition at line 49 of file layers.f90.

#### 5.13.2.2 value

```
type(layer), pointer layers::node_layer::value
```

Pointer to a layer.

Definition at line 48 of file layers.f90.

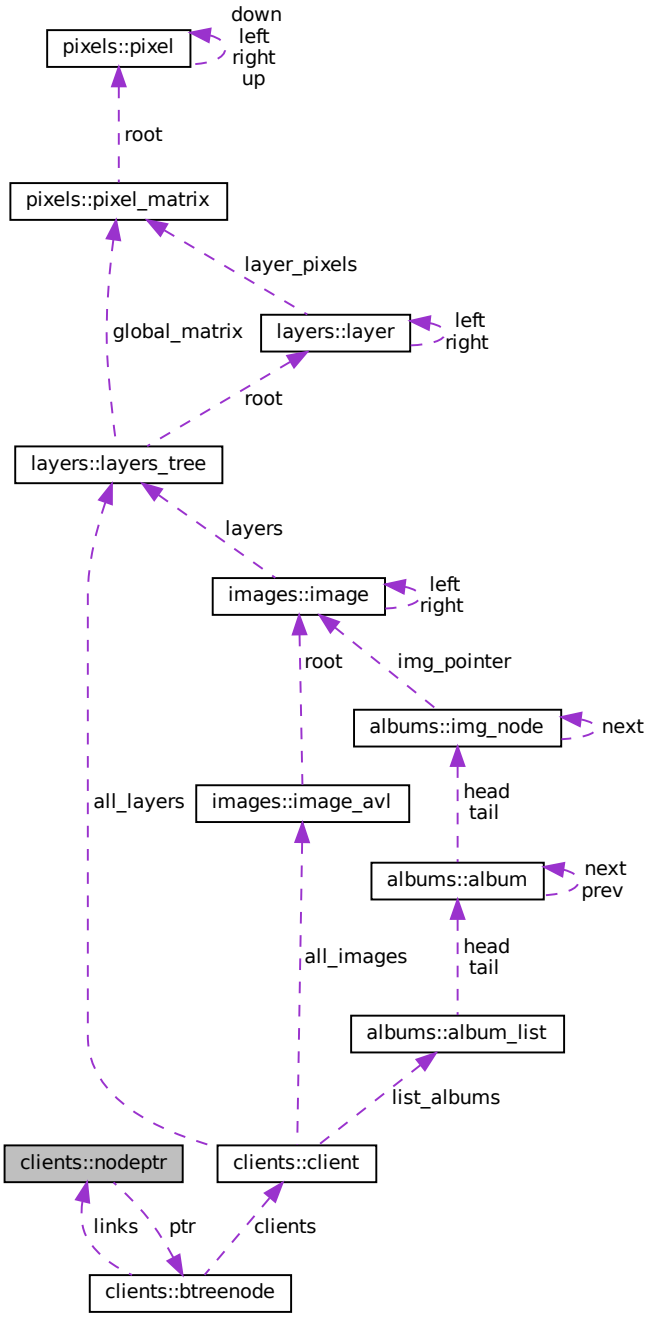
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90

## 5.14 clients::nodeptr Type Reference

Type to represent a node pointer.

Collaboration diagram for clients::nodeptr:




## Public Attributes

- type(btreenode), pointer ptr => null()

  *Pointer to a B-tree node.*

### 5.14.1 Detailed Description

Type to represent a node pointer.

Definition at line 24 of file clients.f90.

### 5.14.2 Member Data Documentation

#### 5.14.2.1 ptr

```
type(btreenode), pointer clients::nodeptr::ptr => null()
```

Pointer to a B-tree node.

Definition at line 25 of file clients.f90.

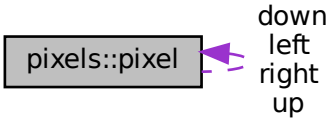
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90

## 5.15 pixels::pixel Type Reference

Type to represent a pixel.

Collaboration diagram for pixels::pixel:



## Public Attributes

- integer id

  *Unique identifier for the pixel.*
- integer x
- integer y

  *Coordinates of the pixel.*
- logical on = .FALSE.

  *Value of the pixel.*
- character(len=7) color

  *Color of the pixel.*
- type(pixel), pointer up => null()

  *Pointer to the pixel above.*
- type(pixel), pointer down => null()

  *Pointer to the pixel below.*
- type(pixel), pointer right => null()

  *Pointer to the pixel on the right.*
- type(pixel), pointer left => null()

  *Pointer to the pixel on the left.*

### 5.15.1 Detailed Description

Type to represent a pixel.

Definition at line 12 of file pixels.f90.

### 5.15.2 Member Data Documentation

#### 5.15.2.1 color

```
character(len=7) pixels::pixel::color
```

Color of the pixel.

Definition at line 16 of file pixels.f90.

#### 5.15.2.2 down

```
type(pixel), pointer pixels::pixel::down => null()
```

Pointer to the pixel below.

Definition at line 18 of file pixels.f90.

#### 5.15.2.3 id

```
integer pixels::pixel::id
```

Unique identifier for the pixel.

Definition at line 13 of file pixels.f90.

#### 5.15.2.4 left

```
type(pixel), pointer pixels::pixel::left => null()
```

Pointer to the pixel on the left.

Definition at line 20 of file pixels.f90.

#### 5.15.2.5 on

```
logical pixels::pixel::on = .FALSE.
```

Value of the pixel.

Definition at line 15 of file pixels.f90.

#### 5.15.2.6 right

```
type(pixel), pointer pixels::pixel::right => null()
```

Pointer to the pixel on the right.

Definition at line 19 of file pixels.f90.

#### 5.15.2.7 up

```
type(pixel), pointer pixels::pixel::up => null()
```

Pointer to the pixel above.

Definition at line 17 of file pixels.f90.

#### 5.15.2.8 x

```
integer pixels::pixel::x
```

Definition at line 14 of file pixels.f90.

#### 5.15.2.9 y

```
integer pixels::pixel::y
```

Coordinates of the pixel.

Definition at line 14 of file pixels.f90.

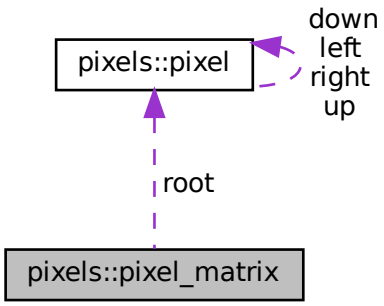
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90

## 5.16 pixels::pixel_matrix Type Reference

Type to represent a matrix of pixels.

Collaboration diagram for pixels::pixel_matrix:



### Public Member Functions

- procedure insert
- procedure self_print
- procedure search_row
- procedure search_column
- procedure node_exists
- procedure insert_column_header
- procedure insert_row_header
- procedure insert_in_row
- procedure insert_in_column
- procedure print_headers
- procedure get_value
- procedure gen_matrix
- procedure get_node
- procedure graph_pixels
- procedure global_m_dot

### Public Attributes

- type(pixel), pointer root => null()

  *Pointer to the root of the matrix.*
- integer width = 0

  *Width of the matrix.*
- integer height = 0

  *Height of the matrix.*

## 5.16.1 Detailed Description

Type to represent a matrix of pixels.

Definition at line 23 of file pixels.f90.

## 5.16.2 Member Function/Subroutine Documentation

### 5.16.2.1 gen_matrix()

```
procedure pixels::pixel_matrix::gen_matrix
```

Definition at line 39 of file pixels.f90.

#### 5.16.2.2 get_node()

```
procedure pixels::pixel_matrix::get_node
```

Definition at line 40 of file pixels.f90.

#### 5.16.2.3 get_value()

```
procedure pixels::pixel_matrix::get_value
```

Definition at line 38 of file pixels.f90.

#### 5.16.2.4 global_m_dot()

```
procedure pixels::pixel_matrix::global_m_dot
```

Definition at line 42 of file pixels.f90.

#### 5.16.2.5 graph_pixels()

```
procedure pixels::pixel_matrix::graph_pixels
```

Definition at line 41 of file pixels.f90.

#### 5.16.2.6 insert()

```
procedure pixels::pixel_matrix::insert
```

Definition at line 28 of file pixels.f90.

#### 5.16.2.7 insert_column_header()

```
procedure pixels::pixel_matrix::insert_column_header
```

Definition at line 33 of file pixels.f90.

#### 5.16.2.8 insert_in_column()

```
procedure pixels::pixel_matrix::insert_in_column
```

Definition at line 36 of file pixels.f90.

#### 5.16.2.9 insert_in_row()

```
procedure pixels::pixel_matrix::insert_in_row
```

Definition at line 35 of file pixels.f90.

#### 5.16.2.10 insert_row_header()

```
procedure pixels::pixel_matrix::insert_row_header
```

Definition at line 34 of file pixels.f90.

#### 5.16.2.11 node_exists()

```
procedure pixels::pixel_matrix::node_exists
```

Definition at line 32 of file pixels.f90.

#### 5.16.2.12 print_headers()

```
procedure pixels::pixel_matrix::print_headers
```

Definition at line 37 of file pixels.f90.

#### 5.16.2.13 search_column()

```
procedure pixels::pixel_matrix::search_column
```

Definition at line 31 of file pixels.f90.

#### 5.16.2.14 search_row()

```
procedure pixels::pixel_matrix::search_row
```

Definition at line 30 of file pixels.f90.

#### 5.16.2.15 self_print()

```
procedure pixels::pixel_matrix::self_print
```

Definition at line 29 of file pixels.f90.

### 5.16.3 Member Data Documentation

#### 5.16.3.1 height

```
integer pixels::pixel_matrix::height = 0
```

Height of the matrix.

Definition at line 26 of file pixels.f90.

#### 5.16.3.2 root

```
type(pixel), pointer pixels::pixel_matrix::root => null()
```

Pointer to the root of the matrix.

Definition at line 24 of file pixels.f90.

#### 5.16.3.3 width

```
integer pixels::pixel_matrix::width = 0
```

Width of the matrix.

Definition at line 25 of file pixels.f90.

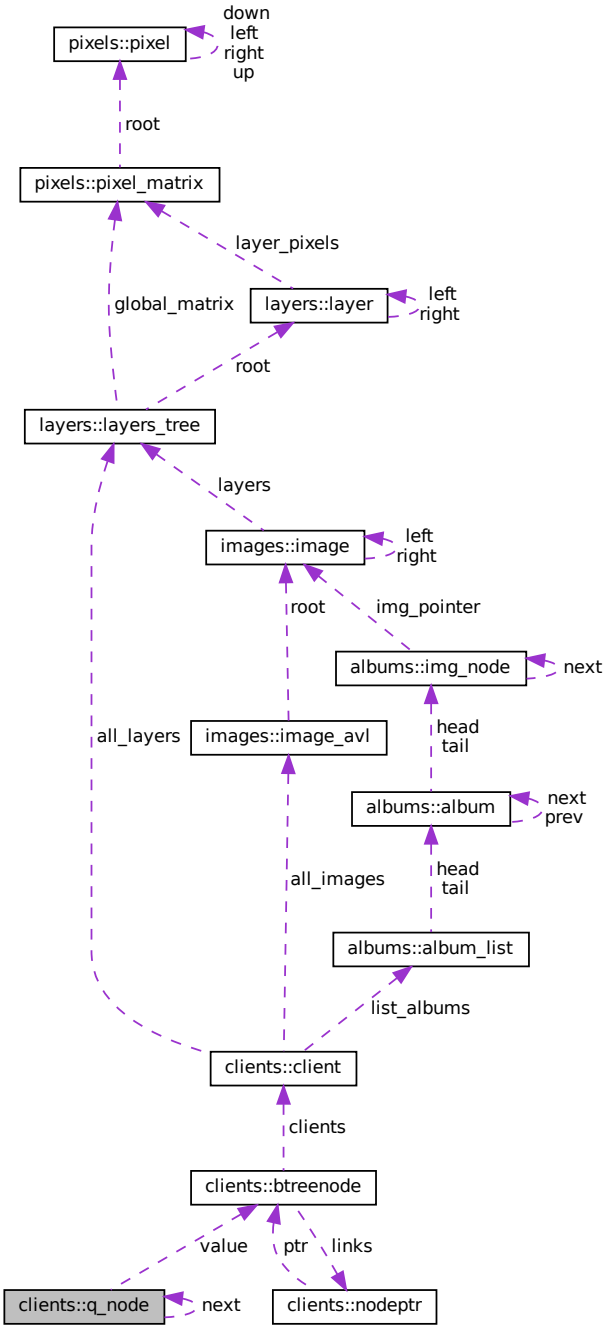
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90

## 5.17 clients::q_node Type Reference

Type to represent a queue node.

Collaboration diagram for clients::q_node:




### Public Attributes

- type(btreenode), pointer value => null()

*Value of the node.*

- type(q_node), pointer next => null()

  *Pointer to the next node.*

### 5.17.1 Detailed Description

Type to represent a queue node.

Definition at line 50 of file clients.f90.

### 5.17.2 Member Data Documentation

#### 5.17.2.1 next

```
type(q_node), pointer clients::q_node::next => null()
```

Pointer to the next node.

Definition at line 52 of file clients.f90.

#### 5.17.2.2 value

```
type(btreenode), pointer clients::q_node::value => null()
```

Value of the node.

Definition at line 51 of file clients.f90.

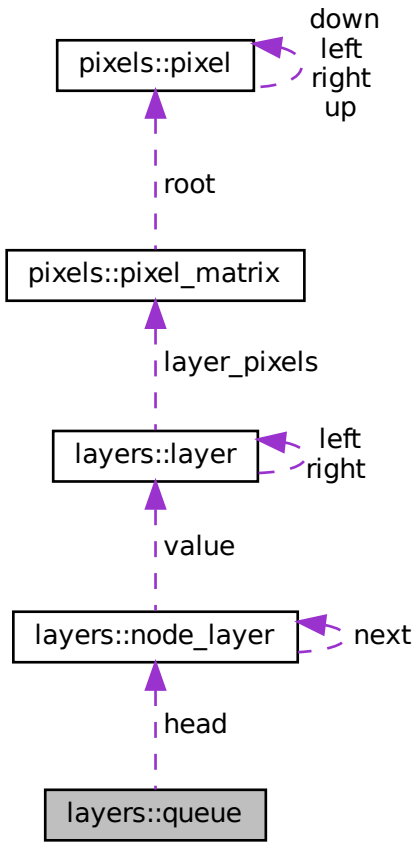
The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90

## 5.18 layers::queue Type Reference

Type to represent a queue of layers.

Collaboration diagram for layers::queue:




### Public Member Functions

- procedure enqueue
- procedure dequeue
- procedure is_empty

### Public Attributes

- type(node_layer), pointer head => null()

  *Pointer to the head of the queue.*

### 5.18.1 Detailed Description

Type to represent a queue of layers.

Definition at line 52 of file layers.f90.

### 5.18.2 Member Function/Subroutine Documentation

#### 5.18.2.1 dequeue()

```
procedure layers::queue::dequeue
```

Definition at line 56 of file layers.f90.

#### 5.18.2.2 enqueue()

```
procedure layers::queue::enqueue
```

Definition at line 55 of file layers.f90.

#### 5.18.2.3 is_empty()

```
procedure layers::queue::is_empty
```

Definition at line 57 of file layers.f90.

### 5.18.3 Member Data Documentation

#### 5.18.3.1 head

```
type(node_layer), pointer layers::queue::head => null()
```

Pointer to the head of the queue.

Definition at line 53 of file layers.f90.

The documentation for this type was generated from the following file:

- /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90

# Chapter 6

# File Documentation

## 6.1 /home/diego/Documents/EDD/Fase 2 Documentation/src/albums.f90 File Reference

Module for albums and images management.

### Data Types

- type albums::img_node

  *Type to store the image node.*
- type albums::album

  *Type to store the album.*
- type albums::album_list

  *Type to store the album list.*

### Modules

- module albums

### Functions/Subroutines

- subroutine albums::gen_album_graph (this, unit)

  *Generates the graph of the albums.*
- subroutine albums::show_album_images (this, id_album)

  *Shows the images of an album.*
- subroutine albums::show_albums (this)

  *Shows the albums.*
- subroutine albums::remove_image_from_album (this, ralbum_id, image_id)

  *Removes an image from an album.*
- subroutine albums::add_image_to_album (this, ialbum_id, image_node)

  *Adds an image to an album.*
- type(image) function, pointer albums::search_in_album (this, salbum_id, image_id)

  *Searches an image in an album.*

- type(album) function, pointer albums::get_album (this, id)

  *Gets an album.*
- subroutine albums::remove_album (this, id)

  *Removes an album.*
- subroutine albums::new_album (this, album_name)

  *Creates a new album.*
- subroutine albums::add_image (this, image_node)

  *Adds an image to an album.*
- subroutine albums::remove_image (this, id)

  *Removes an image from an album.*
- type(image) function, pointer albums::get_image (this, id)

  *Gets an image.*
- subroutine albums::show_images (this)

  *Shows the images of an album.*

### Variables

- integer albums::album_id = 0

  *Global variable to store the album id.*

### 6.1.1 Detailed Description

Module for albums and images management.

**Author**

Diego Cali

**Date**

2024

**Version**

1.0

This module contains the definition of the albums and images management

## 6.2 /home/diego/Documents/EDD/Fase 2 Documentation/src/clients.f90 File Reference

Module to manage clients in a B-tree.

## Data Types

- type clients::client

  *Type to represent a client.*
- type clients::nodeptr

  *Type to represent a node pointer.*
- type clients::btreenode

  *Type to represent a B-tree node.*
- type clients::btree_clients

  *Type to represent a B-tree.*
- type clients::q_node

  *Type to represent a queue node.*
- type clients::client_queue

  *Type to represent a client queue.*

## Modules

- module clients

## Functions/Subroutines

- subroutine clients::amplitude_traversal (this)

  *Subroutine to traverse the tree in amplitude.*
- logical function clients::is_empty (this)

  *Function to check if the queue is empty.*
- type(btreenode) function, pointer clients::dequeue (this)

  *Function to dequeue a node from the queue.*
- subroutine clients::enqueue (this, tree_node)

  *Subroutine to enqueue a node in the queue.*
- subroutine clients::add_client (this, new_client)

  *Subroutine to add a client to the tree.*
- recursive logical function clients::set_value (new_client, pclient, node, child)

  *Recursive function to set the value of the tree.*
- type(btreenode) function, pointer clients::create_node (this, new_client, child)

  *Function to create a node.*
- subroutine clients::insert_node (pclient, pos, node, child)

  *Subroutine to insert a node.*
- subroutine clients::split_node (new_client, pclient, pos, node, child, new_node)

  *Subroutine to split a node.*
- subroutine clients::traversal (this, node)

  *Subroutine to traverse the tree.*
- subroutine clients::clients_dot (this, node, unit)

  *Subroutine to print the tree in dot format.*
- recursive type(client) function, pointer clients::search_client (this, node, client_id)

  *Function to search a client in the tree.*
- subroutine clients::delete_client (this, client_id)

  *Subroutine to delete a client from the tree.*
- subroutine clients::delete_client_rec (this, father, temp, client_id)

  *Recursive subroutine to delete a client from the tree.*
- logical function clients::find (this, client_id)

  *Function to find a client in the node.*

### Variables

- integer clients::g_id = 1

  *Global variable to assign the id of the nodes.*

### 6.2.1 Detailed Description

Module to manage clients in a B-tree.

**Author**

Diego Cali

**Date**

2024

**Version**

1.0

This module contains the definition of the clients module, which is used to manage clients in a B-tree. The module contains the definition of the client type, which contains the name, dpi, password, and the albums and layers of the client. The module also contains the definition of the B-tree node type, which contains the clients and links to other nodes. The module contains the definition of the B-tree type, which contains the root of the tree. The module contains the definition of the node pointer type, which is used to traverse the tree. The module contains the definition of the queue node type, which is used to traverse the tree in amplitude. The module contains the definition of the client queue type, which is used to traverse the tree in amplitude. The module contains the definition of the functions and subroutines to add, search, delete, traverse, and print the tree. The module contains the definition of the functions and subroutines to insert, split, and set the value of the tree. The module contains the definition of the functions and subroutines to enqueue, dequeue, and check if the queue is empty

## 6.3 /home/diego/Documents/EDD/Fase 2 Documentation/src/filehandler.f90 File Reference

Module that handles the reading of the json files and the initialization of the data structures.

### Data Types

- type filehandler::fhandler

  *Type that handles the reading of the json files and the initialization of the data structures.*

### Modules

- module filehandler

### Functions/Subroutines

- subroutine filehandler::initialize_admin (this)

  *Initializes the data structures for the admin user.*
- subroutine filehandler::initialize_user (this)

  *Initializes the data structures for the user.*
- subroutine filehandler::set_user (this, user)

  *Sets the user data structures to the fhandler object.*
- subroutine filehandler::read_imgs (this)

  *Reads the images from the json file.*
- subroutine filehandler::read_layers (this)

  *Reads the layers from the json file.*
- subroutine filehandler::read_clients (this)

  *Reads the clients from the json file.*
- subroutine filehandler::read_albums (this)

  *Reads the albums from the json file.*

#### 6.3.1 Detailed Description

Module that handles the reading of the json files and the initialization of the data structures.

**Author**

Diego Cali

**Date**

2024

**Version**

1.0

This module is in charge of reading the json files and initializing the data structures that will be used in the program

## 6.4 /home/diego/Documents/EDD/Fase 2 Documentation/src/images.f90 File Reference

Module that contains the image and image_avl types and their methods.

### Data Types

- type images::image

  *Type that represents a node in the AVL tree.*
- type images::image_avl

  *Type that represents an AVL tree of images.*

## Modules

- module images

## Functions/Subroutines

- subroutine images::delete_img (this, img_id)

  *Method that deletes an image from the tree.*
- recursive type(image) function, pointer images::delete_img_rec (this, temp, img_id)

  *Recursive method that deletes an image from the tree.*
- recursive type(image) function, pointer images::min_child (this, temp)

  *Method that returns the node with the minimum value in the tree.*
- subroutine images::add_layer (this, new_layer)

  *Method that adds a layer to the image.*
- subroutine images::add_img (this, new_image)

  *Method that adds an image to the tree.*
- subroutine images::add_img_rec (this, new_image, tmp)

  *Recursive method that adds an image to the tree.*
- recursive type(image) function, pointer images::search_img (this, temp, img_ig)

  *Method that searches for an image in the tree.*
- type(image) function, pointer images::srl (this, t1)

  *Method that performs a single right rotation.*
- type(image) function, pointer images::srr (this, t1)

  *Method that performs a single left rotation.*
- type(image) function, pointer images::drl (this, tmp)

  *Method that performs a double right rotation.*
- type(image) function, pointer images::drr (this, tmp)

  *Method that performs a double left rotation.*
- integer function images::get_max (this, val1, val2)

  *Method that returns the maximum value between two integers.*
- integer function images::get_height (this, tmp)

  *Method that returns the height of a node.*
- subroutine images::get_dot (this, tmp, unit)

  *Method that generates the dot representation of the tree.*
- subroutine images::get_dot_rec (this, tmp, unit)

  *Recursive method that generates the dot representation of the tree.*
- subroutine images::print_images (this, temp)

  *Method that prints the images in the tree.*
- subroutine images::gen_tree_subtree (this, id_img, unit)

  *Method that generates the dot representation of the tree and its subtree.*
- recursive subroutine images::gen_layer_subtree (current_layer, unit)

  *Recursive method that generates the dot representation of the layers subtree.*
- subroutine images::gen_img_traversal (this, unit, id_img)

  *Method that generates the dot representation of the image traversal.*

### 6.4.1 Detailed Description

Module that contains the image and image_avl types and their methods.

**Author**

Diego Cali

**Date**

2024

**Version**

1.0

This module contains the image and image_avl types and their methods

## 6.5 /home/diego/Documents/EDD/Fase 2 Documentation/src/layers.f90 File Reference

Module to handle layers of a neural network.

### Data Types

- type layers::layer

  *Type to represent a pixel matrix*

- type layers::layers_tree

  *Type to represent a binary tree of layers.*

- type layers::node_layer

  *Type to represent a node of a linked list of layers.*

- type layers::queue

  *Type to represent a queue of layers.*

### Modules

- module layers

## Functions/Subroutines

- subroutine layers::gen_inorder (this, tmp, limit, unit)

  *Subroutine to generate the matrix of pixels of a layer.*
- subroutine layers::gen_postorder (this, tmp, limit, unit)

  *Subroutine to generate the matrix of pixels of a layer.*
- subroutine layers::gen_preorder (this, tmp, limit, unit)

  *Subroutine to generate the matrix of pixels of a layer.*
- subroutine layers::traverse_limited (this, order, limit, unit)

  *Subroutine to traverse the layers tree in a limited way.*
- subroutine layers::enqueue (this, layer_val)

  *Subroutine to add a layer to the layers tree.*
- type(layer) function, pointer layers::dequeue (this)

  *Subroutine to remove a layer from the queue.*
- logical function layers::is_empty (this)

  *Function to check if the queue is empty.*
- subroutine layers::traverse_matrix (this)

  *Subroutine to traverse the layers tree.*
- subroutine layers::add (this, new_layer)

  *Subroutine to add a layer to the layers tree.*
- subroutine layers::add_recursive (this, new_layer, tmp)

  *Subroutine to add a layer to the layers tree recursively.*
- subroutine layers::add_copied_val (this, new_layer)

  *Subroutine to add a layer to the layers tree.*
- type(layer) function, pointer layers::search (this, id_searched)

  *Function to search a layer in the layers tree.*
- subroutine layers::preorder (this, tmp)

  *Subroutine to traverse the layers tree in preorder.*
- subroutine layers::inorder (this, tmp)

  *Subroutine to traverse the layers tree in inorder.*
- subroutine layers::postorder (this, tmp)

  *Subroutine to traverse the layers tree in postorder.*
- subroutine layers::gen_dot (this, tmp, unit)

  *Subroutine to generate the dot file of the layers tree.*
- subroutine layers::gen_dot_recursive (this, tmp, unit)

  *Subroutine to generate the dot file of the layers tree recursively.*
- subroutine layers::max_depth (this)

  *Subroutine to calculate the max depth of the layers tree.*
- recursive integer function layers::max_depth_rec (this, root)

  *Function to calculate the max depth of the layers tree recursively.*
- subroutine layers::leaf_layers (this)

  *Subroutine to traverse the leaf layers of the layers tree.*
- subroutine layers::leaf_layers_rec (this, tmp)

  *Subroutine to traverse the leaf layers of the layers tree recursively.*
- subroutine layers::list_layers (this, option)

  *Subroutine to list the layers of the layers tree.*
- subroutine layers::inorder_print (this, tmp)

  *Subroutine to print the layers of the layers tree in inorder.*

### 6.5.1 Detailed Description

Module to handle layers of a neural network.

**Author**

Diego Cali

**Date**

2024

**Version**

1.0

This module contains the definition of the layers_tree type, which is a binary tree of layers. Each layer has an id, a number of pixels, a pixel_matrix and two pointers to other layers. The module also contains the definition of the pixel_matrix type, which is a matrix of pixels. The module also contains the definition of the node_layer type, which is a node of a linked list of layers. The module also contains the definition of the queue type, which is a queue of layers. The module contains the following subroutines and functions:

## 6.6 /home/diego/Documents/EDD/Fase 2 Documentation/src/pixels.f90 File Reference

Module to handle pixels in a matrix.

### Data Types

- type pixels::pixel

  *Type to represent a pixel.*
- type pixels::pixel_matrix

  *Type to represent a matrix of pixels.*

### Modules

- module pixels

## Functions/Subroutines

- subroutine pixels::insert (this, x, y, value, color)

  *Insert a pixel in the matrix.*
- type(pixel) function, pointer pixels::search_row (this, y)

  *Search a row in the matrix.*
- type(pixel) function, pointer pixels::search_column (this, x)

  *Search a column in the matrix.*
- logical function pixels::node_exists (this, new_node)

  *Check if a node exists in the matrix.*
- type(pixel) function, pointer pixels::insert_row_header (this, y)

  *Insert a row header in the matrix.*
- subroutine pixels::insert_in_row (this, new_node, row_header)

  *Insert a pixel in a row.*
- type(pixel) function, pointer pixels::insert_column_header (this, x)

  *Insert a column header in the matrix.*
- subroutine pixels::insert_in_column (this, new_node, column_header)

  *Insert a pixel in a column.*
- subroutine pixels::print_headers (this)

  *Print the headers of the matrix.*
- logical function pixels::get_value (this, x, y)

  *Get the value of a pixel.*
- type(pixel) function, pointer pixels::get_node (this, x, y)

  *Get the node in the matrix.*
- subroutine pixels::self_print (this)

  *Print the matrix.*
- subroutine pixels::graph_pixels (this, unit)

  *Generate a graph of the pixels.*
- subroutine pixels::gen_matrix (this, g_matrix)

  *Generate a matrix of pixels.*
- subroutine pixels::global_m_dot (this, unit)

  *Generate a graph of the pixels.*

## Variables

- integer pixels::id = 0

  *Unique identifier for each pixel.*

### 6.6.1 Detailed Description

Module to handle pixels in a matrix.

**Author**

Diego Cali

**Date**

2024

**Version**

1.0

This module is used to handle pixels in a matrix. It is possible to insert, search, print and generate a graph of the pixels.

# Index