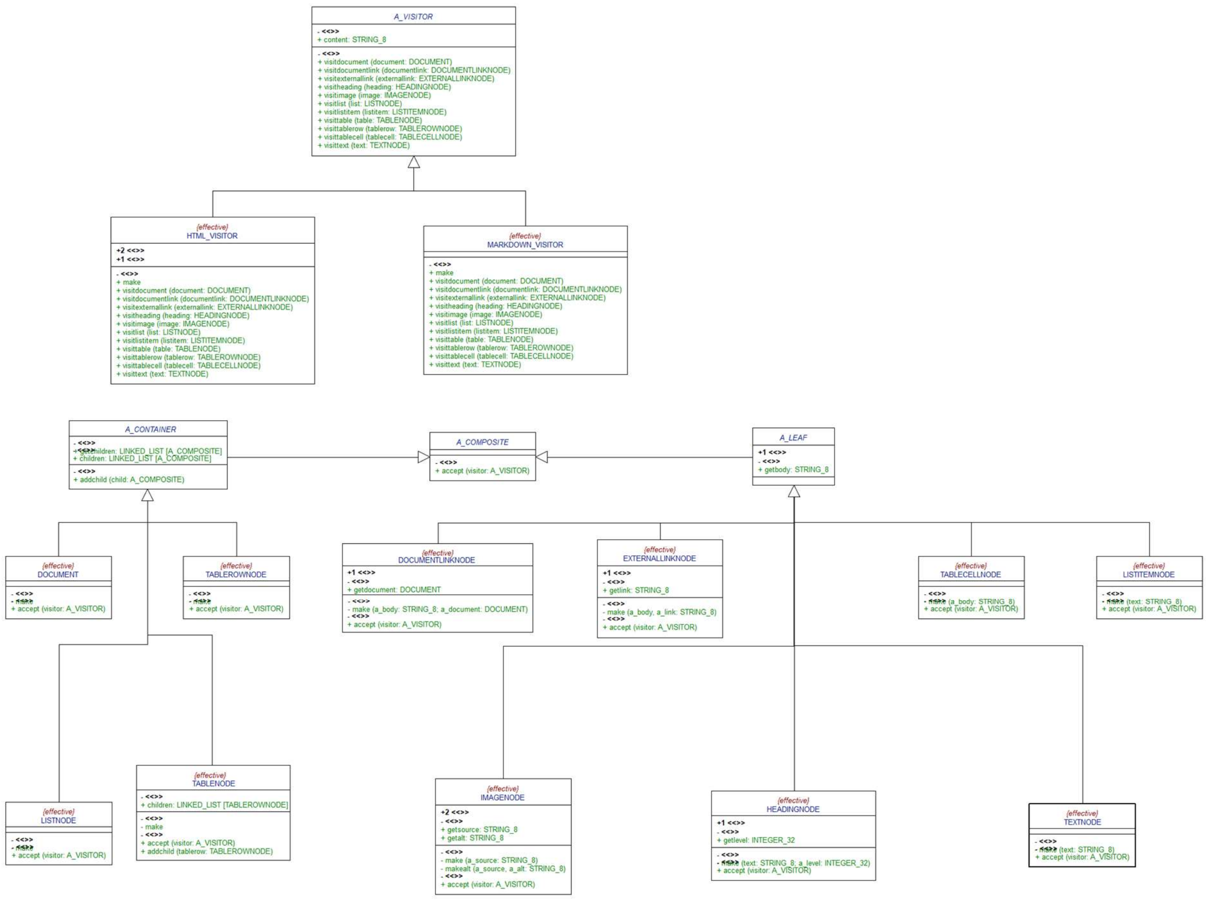
**API Design**


A_VISITOR
- <<>>
+ content: STRING_8
- <<>>
+ visitdocument (document: DOCUMENT)
+ visitdocumentlink (documentlink: DOCUMENTLINKNODE)
+ visitexternallink (externallink: EXTERNALLINKNODE)
+ visitheading (heading: HEADINGNODE)
+ visitimage (image: IMAGENODE)
+ visitlist (list: LISTNODE)
+ visitlistitem (listitem: LISTITEMNODE)
+ visittable (table: TABLENODE)
+ visittablerow (tablerow: TABLEROWNODE)
+ visittablecell (tablecell: TABLECELLNODE)
+ visittext (text: TEXTNODE)
{effective}
HTML_VISITOR
+2 <<>>
+1 <<>>
- <<>>
+ make
+ visitdocument (document: DOCUMENT)
+ visitdocumentlink (documentlink: DOCUMENTLINKNODE)
+ visitexternallink (externallink: EXTERNALLINKNODE)
+ visitheading (heading: HEADINGNODE)
+ visitimage (image: IMAGENODE)
+ visitlist (list: LISTNODE)
+ visitlistitem (listitem: LISTITEMNODE)
+ visittable (table: TABLENODE)
+ visittablerow (tablerow: TABLEROWNODE)
+ visittablecell (tablecell: TABLECELLNODE)
+ visittext (text: TEXTNODE)
{effective}
MARKDOWN_VISITOR
- <<>>
+ make
+ visitdocument (document: DOCUMENT)
+ visitdocumentlink (documentlink: DOCUMENTLINKNODE)
+ visitexternallink (externallink: EXTERNALLINKNODE)
+ visitheading (heading: HEADINGNODE)
+ visitimage (image: IMAGENODE)
+ visitlist (list: LISTNODE)
+ visitlistitem (listitem: LISTITEMNODE)
+ visittable (table: TABLENODE)
+ visittablerow (tablerow: TABLEROWNODE)
+ visittablecell (tablecell: TABLECELLNODE)
+ visittext (text: TEXTNODE)
A_CONTAINER
- <<>>
+ getchildren: LINKED_LIST [A_COMPOSITE]
+ children: LINKED_LIST [A_COMPOSITE]
- <<>>
+ addchild (child: A_COMPOSITE)
A_COMPOSITE
- <<>>
+ accept (visitor: A_VISITOR)
A_LEAF
+1 <<>>
- <<>>
+ getbody: STRING_8
{effective}
DOCUMENT
- <<>>
- make
+ accept (visitor: A_VISITOR)
{effective}
TABLEROWNODE
- <<>>
- make
+ accept (visitor: A_VISITOR)
{effective}
DOCUMENTLINKNODE
+1 <<>>
- <<>>
+ getdocument: DOCUMENT
- <<>>
- make (a_body: STRING_8; a_document: DOCUMENT)
- <<>>
+ accept (visitor: A_VISITOR)
{effective}
EXTERNALLINKNODE
+1 <<>>
- <<>>
+ getlink: STRING_8
- <<>>
- make (a_body, a_link: STRING_8)
- <<>>
+ accept (visitor: A_VISITOR)
{effective}
TABLECELLNODE
- <<>>
- make (a_body: STRING_8)
+ accept (visitor: A_VISITOR)
{effective}
LISTITEMNODE
- <<>>
- make (text: STRING_8)
+ accept (visitor: A_VISITOR)
{effective}
LISTNODE
- <<>>
- make
+ accept (visitor: A_VISITOR)
{effective}
TABLENODE
- <<>>
+ children: LINKED_LIST [TABLEROWNODE]
- <<>>
- make
- <<>>
+ accept (visitor: A_VISITOR)
+ addchild (tablerow: TABLEROWNODE)
{effective}
IMAGENODE
+2 <<>>
- <<>>
+ getsource: STRING_8
+ getalt: STRING_8
- <<>>
- make (a_source: STRING_8)
- makealt (a_source, a_alt: STRING_8)
- <<>>
+ accept (visitor: A_VISITOR)
{effective}
HEADINGNODE
+1 <<>>
- <<>>
+ getlevel: INTEGER_32
- <<>>
- make (text: STRING_8; a_level: INTEGER_32)
+ accept (visitor: A_VISITOR)
{effective}
TEXTNODE
- <<>>
- make (text: STRING_8)
+ accept (visitor: A_VISITOR)

As a first step, we're using the Composite design pattern for the tree-like document structure. This seems to be a reasonable choice as the document is going to be built by different containing elements and leaves.
The classes used are the A_COMPONENT as the root of the whole pattern, the A_CONTAINER as a containing element and the A_LEAF for all elements not containing any other elements. The effective classes used for the required elements inherit from either A_CONTAINER or A_LEAF.

As a second step, to create some specific markdown output, we're using the visitor pattern to traverse the whole document. This way, to add another output, only another visitor must be implemented. The classes used there are the A_VISITOR class which defines the interface for effective implementations of the markup language visitors. The HTML_VISITOR is part of the required implementation, the MARKDONW_VISITOR will be implemented optionally if there's time enough. They get accepted by the implemented composite classes.