OMAS Language v1.0

¿Qué es OMAS?

OMAS es el acrónimo de Object Model App Skeleton

Tiene como objetivo modelar aplicaciones de forma sencilla con muy poco código y portables a varios lenguajes.

La sintaxis de OMAS es similar a la de python

Sintaxis básica

**Comentarios:**

Un comentario empieza con # + texto

#Esto es un comentario

**Fin de línea:**

Solamente un salto de línea

**Bloques de código:**

Un bloque de código se reconoce por la identación

object User:

uid \_id

string name

**Inicio del documento**

Un documento comienza con la palabra project <ProjectName>:

project MyApp:

Con esto el compilador detecta a cual proyecto pertenece el formulario y lo genera en la ruta configurada por el project.json

El contenido del proyecto va tabulado.

**Declarar un objeto:**

Los objetos en OMAS son la pieza fundamental del desarrollo, en base a una simple estructura de datos se genera un API con la definición del objeto y funciones CRUD, además de generarse también un test y una estructura para bases de datos (Tabla o documento) y una serie de formularios HTML5.

Dentro de un Objeto se podrá embeber alguna función con lógica especifica del objeto que será traducida al lenguaje de backend seleccionado.

Ej.

**object** <ObjectName>:

#Size Specific

Type Size PropertieName

Or

#Dynamic size

Type PropertieName

El tamaño especifico define el tamaño del campo que ocupará esa propiedad, en DB se creará un row con ese tamaño y en lenguajes como Go,C++, etc. Se crearán los objetos con estas características.

El tamaño dinámico deja al compilador calcular el tamaño del dato en base a el nombre de la variable y el tipo.

Propiedades prefabricadas

El lenguaje al estar pensado en agilizar el proceso de desarrollo posee ya algunas propiedades reutilizables con un comportamiento, que son recurrentes en casi cualquier aplicación, tales como un \_id, \_email, \_password.

Las propiedades prefabricadas se llaman anteponiendo \_ al nombre de la propiedad:

**object** Usuario:

\_id

string 50 nombre

**Las propiedades prefabricadas son:**

**\_id:** Genera un id único en el backend y en la base de datos

**\_email**: Genera una validación en el backend y fronted para que cumpla con los requisitos básicos de una dirección de email

**\_password**: Genera una función de un solo camino en la que se encripta el password en el constructor del objeto y se almacena el hash creado.

Proyecto de ejemplo:

project Myapp:

#clase usuario

object User:

uid \_id

string 50 name

string 13 password

bool isLogged

public def isOk(self,tu):

**Representation**

HTML5 Forms

**Storage**

Mysql

Mongo

**API**

**TEST**

**Logic**

PHP

Python

Node

Go

**object**