Drakobit Foundation [EMCH91]

<https://emch91p.wixsite.com/drakobit/quienes-somos>

**CEO: Edgar Mauricio Chara Hurtado.**

**Correo electrónico:** [**charalucifer@gmail.com**](mailto:charalucifer@gmail.com)

**Instagram : @emchsocial**

[Larry] programming language.  
(Reference guide) .

# Fast notes.

## Introduction:

Larry is a digital programming language, with a human interface and not a machine approach.

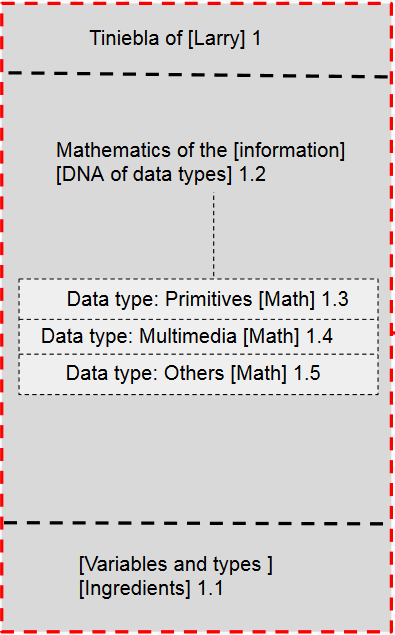
* Mathematics is the fundamental core of [Larry [digital programming language]].
  + Definition: Mathematics, is the source code of a system [X]. It's not numerology, symbology and geometry. The logical and physical order of the elements allows us to find the harmony of a system [X].

## Variables and types :

[Larry] Programming language [Variables and types] [1]

* + Ingredients: It is the raw material [Type of data]. Basic element, which can be combined with other elements, to generate a recipe [[Fine plate], [Digital application]].

[Variables and types][Ingredients] .



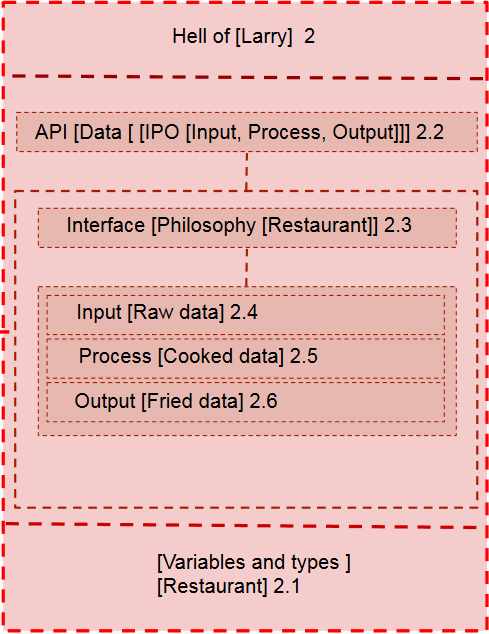
* + Restaurant: The data API of [Larry [Programming Language]] implements an Interface [IPO [Input - Process - Output]]. The grace of the interface [IPO] is, the philosophy [Restaurant].

Input: [Raw data] [Ready to be stored in RAM memory [data storage, object oriented].

Process: [Cooked data] [Ready to be combined with other ingredients [Declared data] [Product catalog]].

Output: [Fried data] [Ready to be consumed] [Defined data].

[Variables and types][Restaurant] .



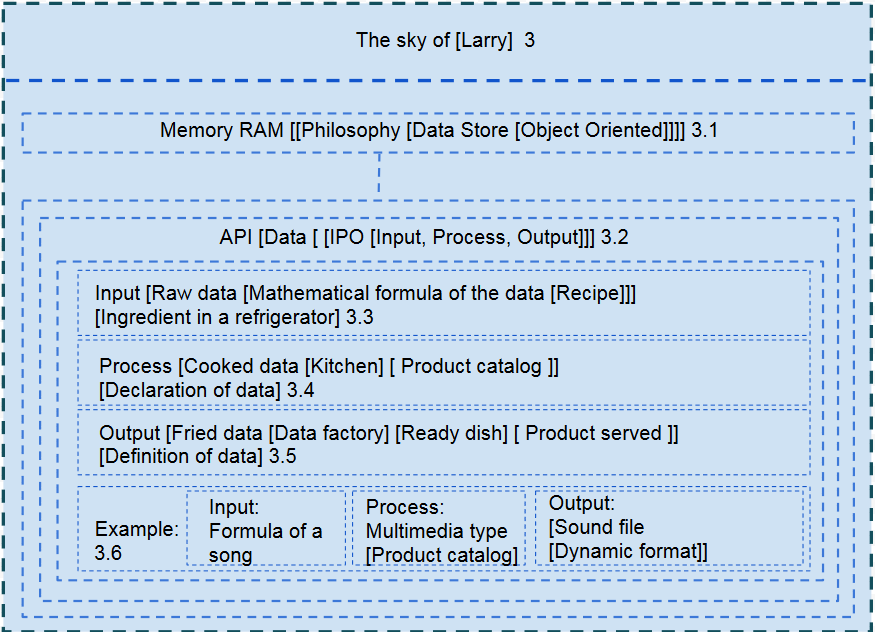
* + RAM Memory [Data store [Object Oriented]]: The data store in [RAM] memory of [Larry [Programming Language]] implements an interface [IPO [Input - Process - Output]]. The grace of the interface [IPO] is, the philosophy [Oriented to objects].

Each project with [Larry [Digital Programming Language]], creates an instance of the data store [Object Oriented] in the RAM memory, to eliminate the manual manipulation of the RAM Memory.

Input [Raw data [Mathematical formula of the data [Recipe]]],  
 [Ingredient in a refrigerator].

Process [Cooked data [Kitchen] [ Product catalog ]][Declaration of data].

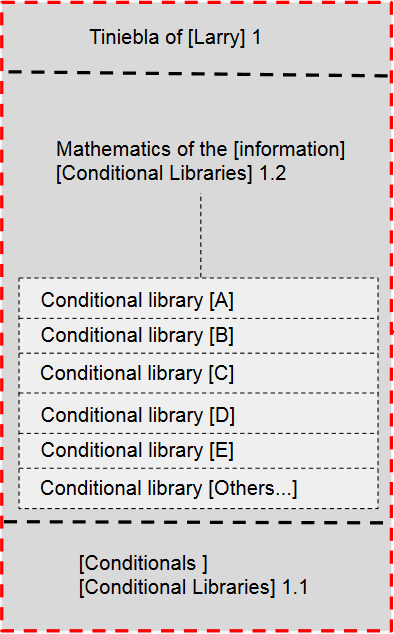
Output [Fried data [Data factory] [Ready dish] [ Product served ]],  
 [Definition of data].



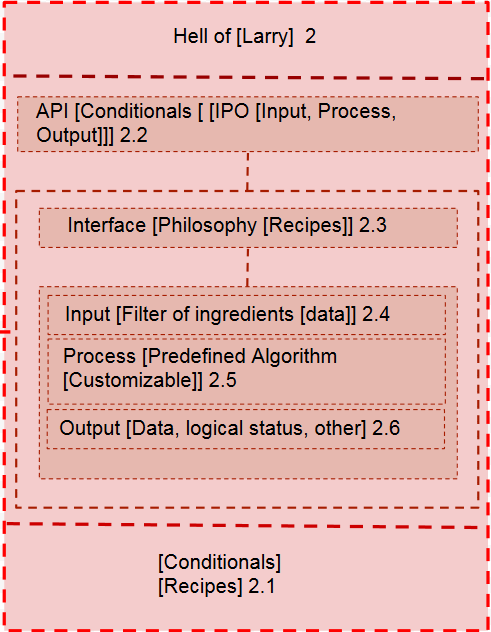
## Conditionals:

[Larry] Programming language [Conditionals] [2]

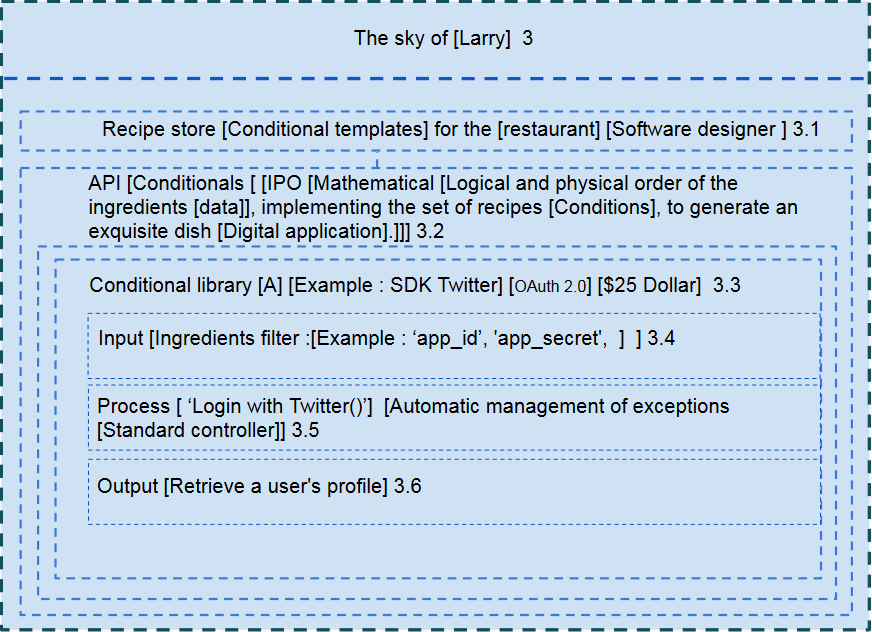
* + [Larry] is a digital programming language, with human interface - no machine.
  + The fundamental basis of [Larry [Programming Language]], is the Mathematical [Non-numerical, non-symbolic, non-geometric]. The math that implements [Larry], is the logical-physical order of the elements, to find the harmony of a system [X].
  + [Larry [Digital Programming Language]], is object oriented and the grace is to eliminate the complexity of the user - developer, by eliminating the machine approach.
  + The Human Interface, which implements [Larry] [Digital Programming Language], is through a philosophy [Restaurant]. This is how, the variables and data types, represent for [Larry] cooking ingredients.
  + The power of [Larry [Digital Programming Language]], is to take full advantage of the logic of the user - developer, stripping the technical details. The friendly human interface of [Larry] implements the conditional logic in a different way. Traditional programming languages, such as [C ++, Java, PHP, JavaScript, among others], use the traditional structure [if-else], so that a digital application can make decisions.
  + [Larry [Digital Programming Language]], implements a new term [Recipes [Conditional Templates]]. The recipes define a set of conditions that facilitate the task of creating an [Exquisite dish [Digital application]]. The recipes are stored in a store [Under dynamic licensing [Free templates and Payment Templates]]. The recipe store [Conditional Templates], from [Larry [Digital Programming Language]], offers the standard package, included in the [Larry] DEVELOPMENT KIT.
  + The user - developer, integrates the project [Exquisite dish [Digital application]], one or multiple [Conditional templates [Set of ingredients, encapsulated and integrated, by default]]. The user - developer, does not incur the redundancy [Create conditions manually, through the traditional conditional structure [[if- else], [do-while], others ...].
  + The math of [Larry [Digital Programming Language]], allows the user - developer:  
      
    Sort the physical-logical elements of the [recipe [conditional template]], to customize the [exquisite dish [Digital application]], without worrying about the low level technical details. Each Recipe [Conditional Template], integrates one or several standard controllers [To handle exceptions and errors, implementing the interface [Conditionals - IPO] [Input - Process - Output]]. The function of the standard controllers [To handle exceptions and errors], is to delegate the responsibility to the manufacturer of the recipe [Conditional Template] and create for the user - developer [Chef], a high level customization interface.
  + [Conditional Libraries].



* + [Conditional] [Recipes].

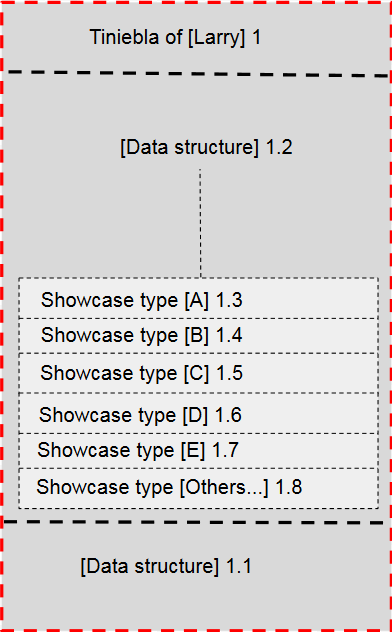


* + [Exquisite dish [Digital application]].

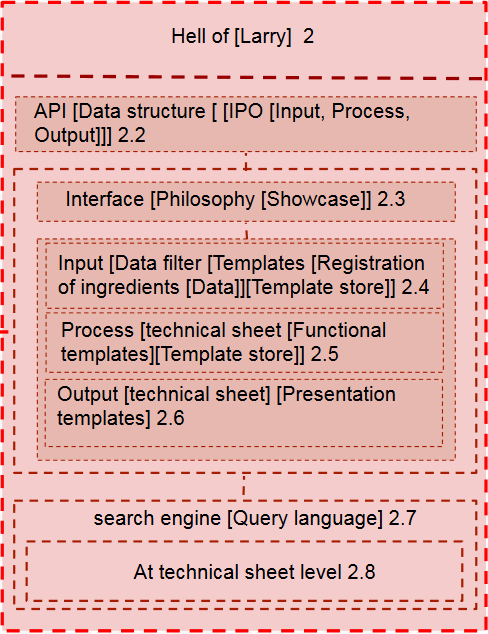


## Data structure and search engine [Query language]:

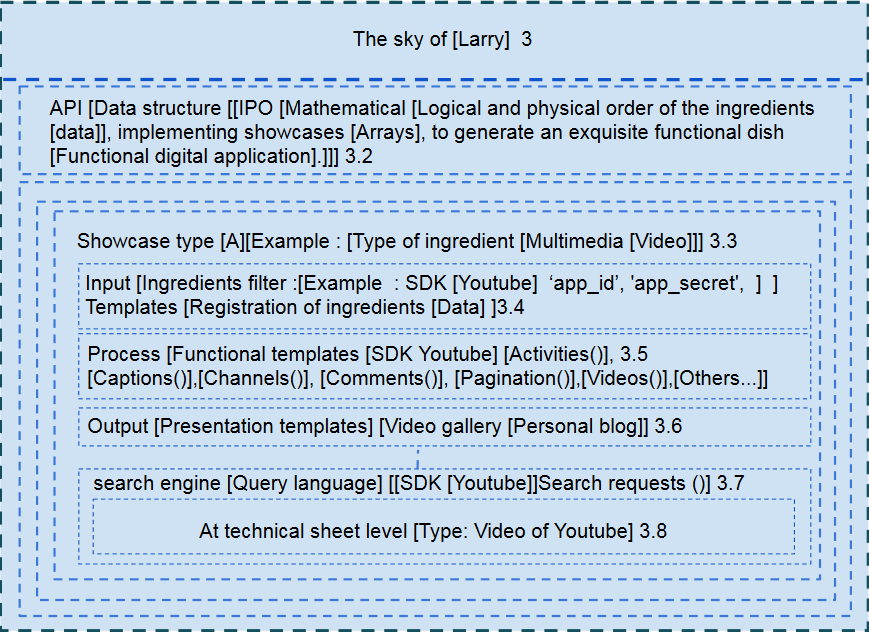
* + [Larry [Digital Programming Language]], places at the disposal of [Chef [Developer User]] a dynamic data structure, through display cases of different types.
  + The grace of the dynamic data structure is the philosophy [Showcase]. The showcase is adjusted automatically, to the type of ingredient [Data type].
  + The dynamic data structure [Showcase] is controlled through the API [IPO [Input - Process - Output]].
  + Input: [Larry [Digital programming language]], makes available to the Chef [User - developer], registration templates with a [data filter], according to the type of showcase.
  + Process: [Larry [Digital Programming Language]], places at the disposal of the Chef [User - developer], functional templates with a [filter: technical sheet] of the ingredient.
  + Output: [Larry [Digital Programming Language]], makes available to the Chef [User - developer], presentation templates with a [filter: technical sheet] of the ingredient.
  + Query Engine: [Larry [Digital Programming Language]], places at the disposal of the Chef [User - developer], a search engine with [Query Language] [At technical sheet level].
  + [Data structure [Dynamic]]



* + Data structure [Interface [Philosophy [Showcase]]]

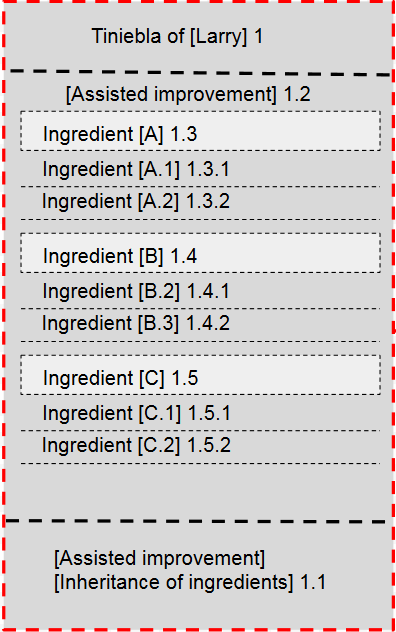


* + Exquisite functional dish [Functional digital application]

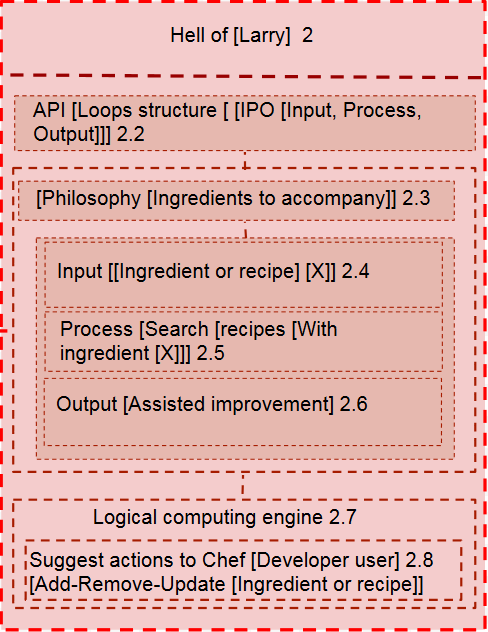


## Loops:

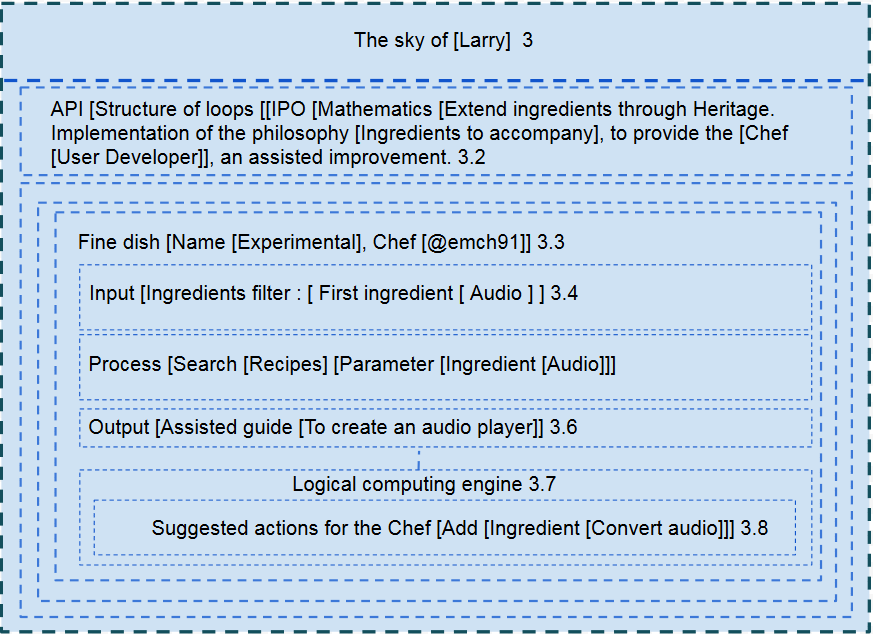
* + [Larry] [Digital programming language], provides the Chef [User developer], through [Loops] a assisted improvement.
  + Each ingredient [Data Type] of [Larry [Programming Language]] is part of a family [Inheritance of Ingredients [Data Types]].
  + The grace of the structure [Loops], is the philosophy [Ingredients to accompany].
  + Through the [IPO [Input-Process-Output]] Interface, the [Logical Computing Engine] creates the guided assistance.
  + Input: [[Ingredient or recipe] [X]].
  + Process: Process [Search [recipes [With ingredient [X]]].
  + Output [Assisted improvement].
  + [Assisted improvement][Inheritance of ingredients]



* + Logical computing engine.

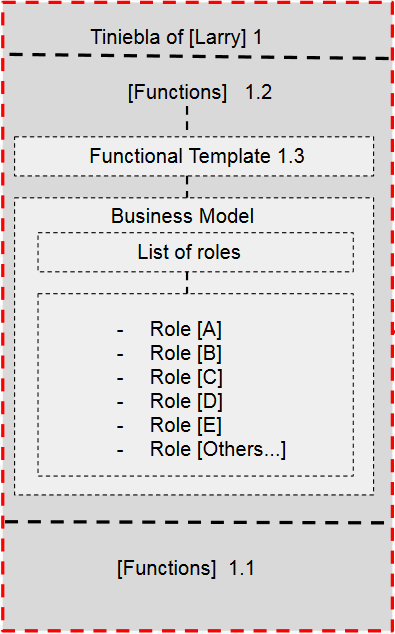


* + Assisted improvement.

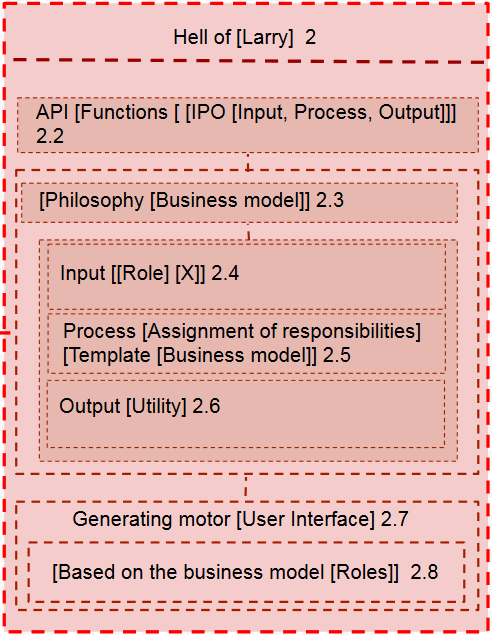


## Functions:

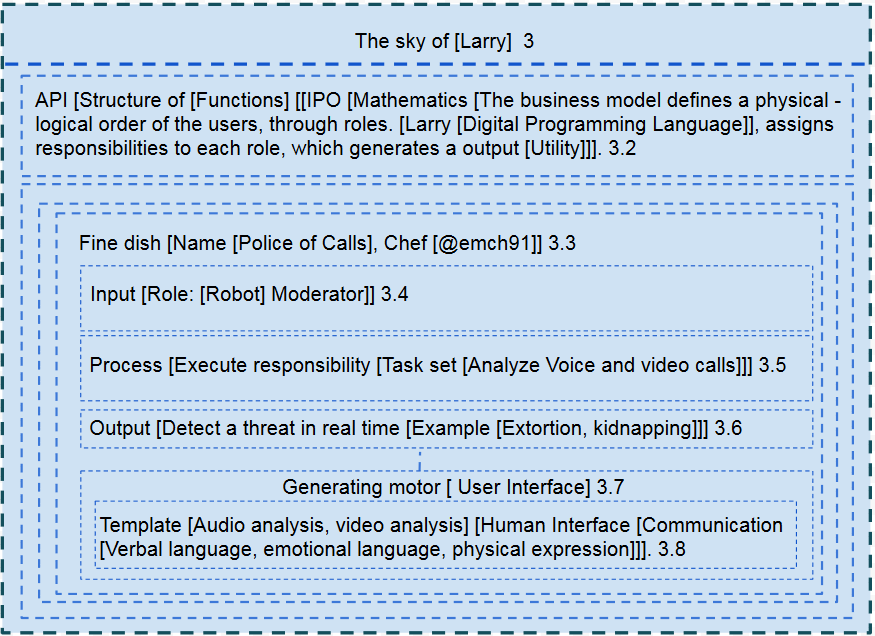
* + [Larry [Digital Programming Language]], creates the structure of [Functions], implementing templates [Business Model]. The Template [Business Model] defines roles by default.
  + [Larry [Digital Programming Language]], implements the Mathematical [Physical and logical order of the roles] and through the API [IPO [Functions] [Input-Process-Exit]], assigns responsibilities, according to the business model.
  + It is like this:
  + Input: Role [X].
  + Process: Assignment of responsibilities, according to the template [Business Model].
  + Output: Utility.
  + Generator Engine [User Interface]: [Larry [Digital Programming Language]], through the IPO Interface [Functions], generates the [User Interface [Based on the business model]]. The user interface, generated by [Larry], is stripped of graphic elements and is a clean communication mechanism, between the User and the exquisite dish [Digital Application].
  + Functions



* + API [Functions [IPO [Input, Process, Output]]].

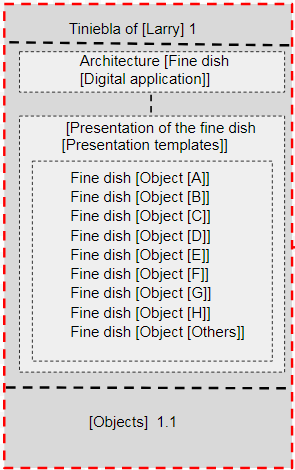


* + API [Functions [IPO [Input, Process, Output]]].

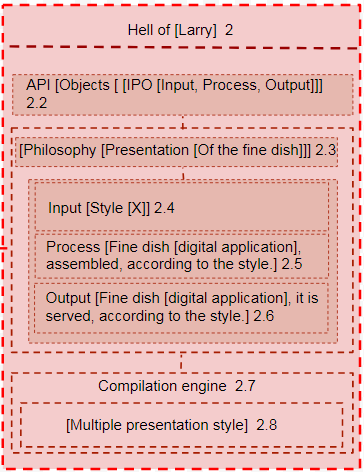


## Objects:

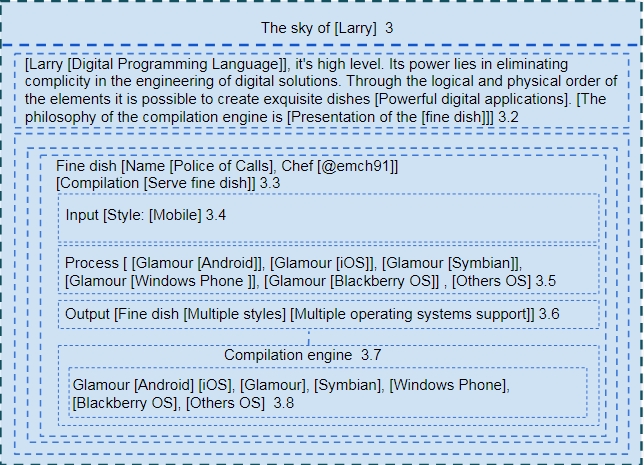
* + [Larry [Digital Programming Language]], it's high level. [Larry], eliminates complicity, when creating digital solutions. Through the philosophy [Restaurant], the user developer [@chef], implements predefined templates [Recipes [Conditional Algorithms]], to create fine dishes [Digital applications].
  + [Larry], provides the user with a [Guided Improvement], through the logical computing engine, which implements [Ingredient Heritage], to suggest [Accompanying Ingredients] and also suggest [Accompanying Recipes]. The philosophy [Accompanying Ingredients], allows the user developer [Chef], to create powerful applications in time records and to scale the functionality within the context of the fine dish, with ease.
  + [Larry], allows the developer user [@chef], to create a user interface [No graphic elements [Functional Focus]], automatically by means of [Predefined Functional Templates]. Each functional template implements one or several business models. The business models define user roles and through the interface [IPO [Input - Process - Output]], it is possible to assign each role [Responsibilities]. The clean user interface, which [Larry] generates, automatically, allows the compilation engine to generate multiple [Styles].
  + Each style is a [Glamor [natural charm that fascinates]], for the [Fine plane architecture [Digital application]]. The advantage for the user developer [@chef] or developer [@restaurante], is the pure Mathematics of [Larry], stripped of the [Numerology], [Symbology], [Geometry]. Mathematics is the source code of a system [X], the logical and physical order of the elements [Ingredients], allows to create recipes [Solution Algorithms], to create a system [Fine dish [Great digital application]].
  + Objects



* + Multiple presentation style

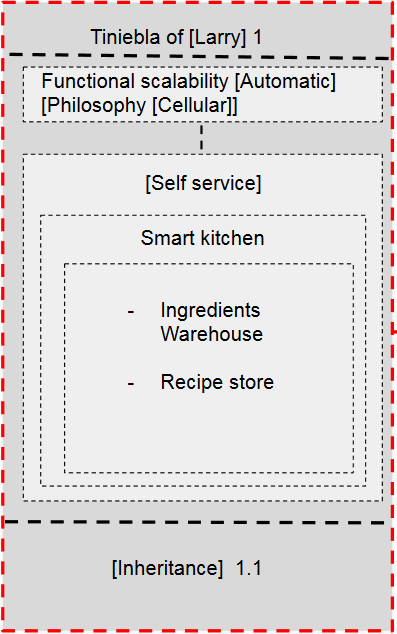


* + Compilation engine [Multiple output]

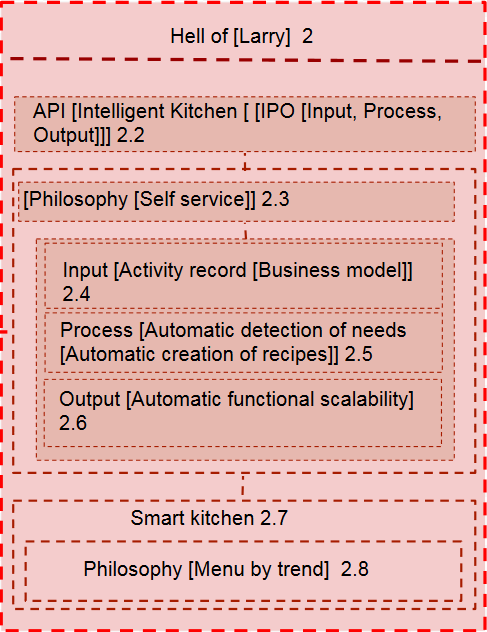


## Heritage:

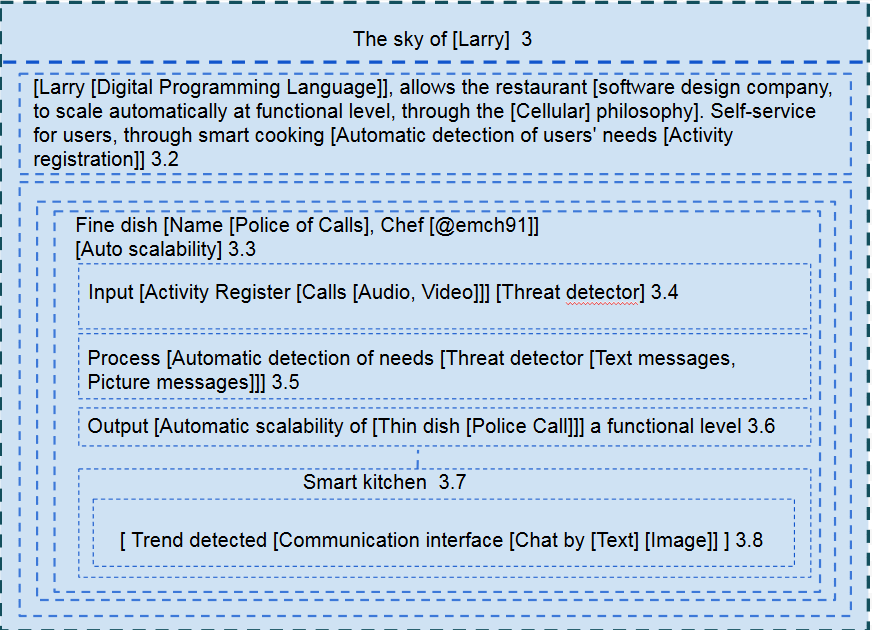
* + [Larry [Digital Programming Language]], Allows automatic scalability [a functional level], through [Intelligent Cooking [Intelligent Compilation]]. [Larry], must have an economy, which allows you to create solutions without depending on the user developer [@chef]. The economy of [Larry], is the store of [Ingredients] [Type of data] and the recipe store. Through the Interface [IPO [Input - Process - Output]], it is possible that the restaurant [Home developer software], provides a Self-Service [Auto Scalability [a functional level]].
  + [Larry [Digital Programming Language]], allows automatic scalability, through [Intelligent Cooking [Intelligent Compilation]]. [Larry], you must have an economy, which allows you to create solutions without depending on the user's developer [@chef]. The economy of [Larry], is the store of [Ingredients] [Type of data] and the store of recipes. Through the [IPO [Input - Process - Output]] interface, it is possible that the [Home developer software] restaurant offers a self-service [Auto Scalability [a functional level]].
  + Entry: [Activity log [Business model]].
  + Process: [Automatic detection of needs [Automatic creation of recipes]].
  + Output: [Automatic scalability [a functional level]].
  + Inheritance:



* + Smart Kitchen [Smart Compiler]:

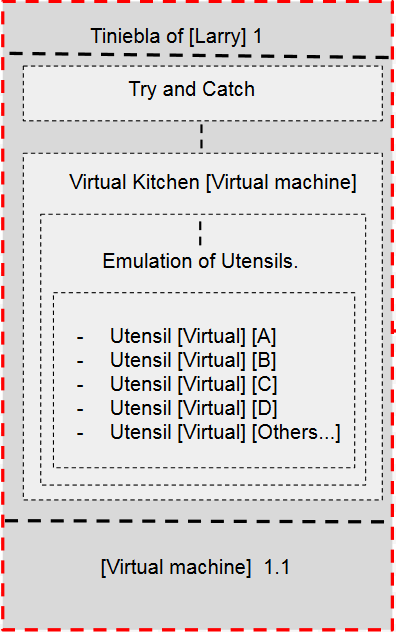


* + Automatic Scalability [Functional level].

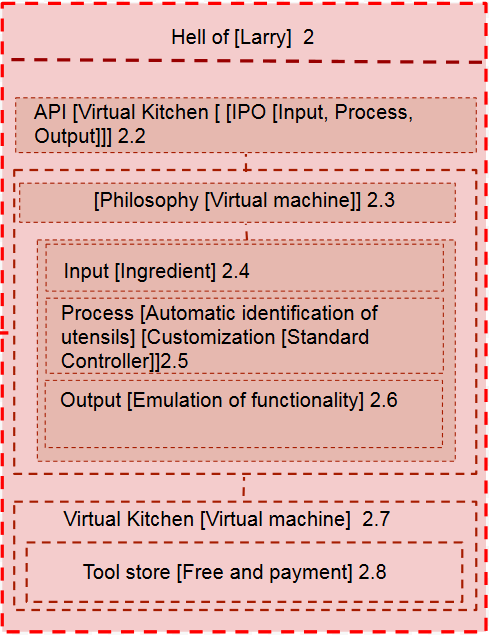


## Try and Catch:

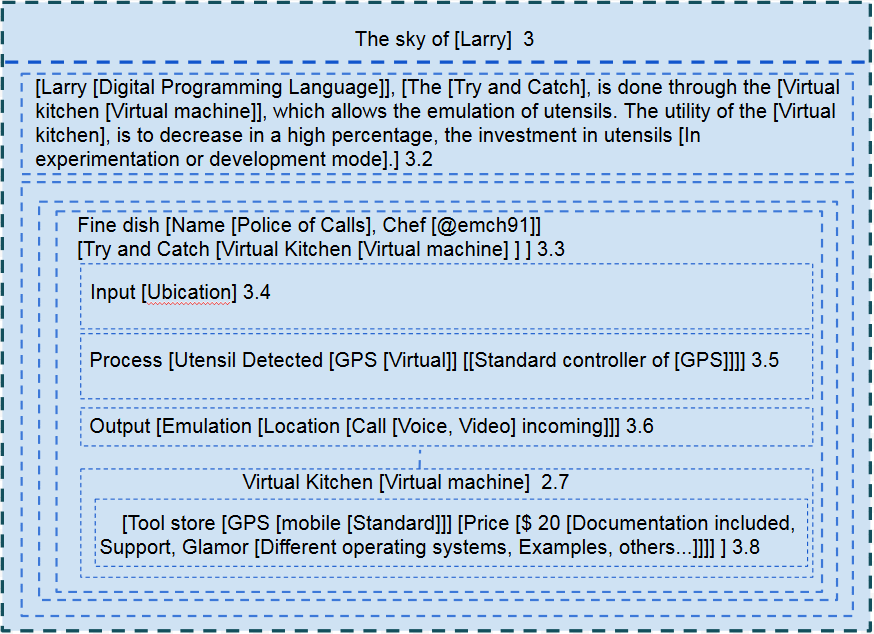
* + [Larry [Digital Programming Language]], [The [Try and Catch], is done through the [Virtual kitchen [Virtual machine]], which allows the emulation of utensils. The utility of the [Virtual kitchen], is to decrease in a high percentage, the investment in utensils [In experimentation or development mode].]
  + Input: [Ingredient].
  + Process: [Automatic identification of utensils] [Customization [Standard Controller]].
  + Output: [Emulation of functionality].
  + Virtual machine.



* + Tool store [Free and payment].



* + Emulation of functionality [Virtual kitchen [Virtual machine]].



## 

## 

## 

## 

## 

## 

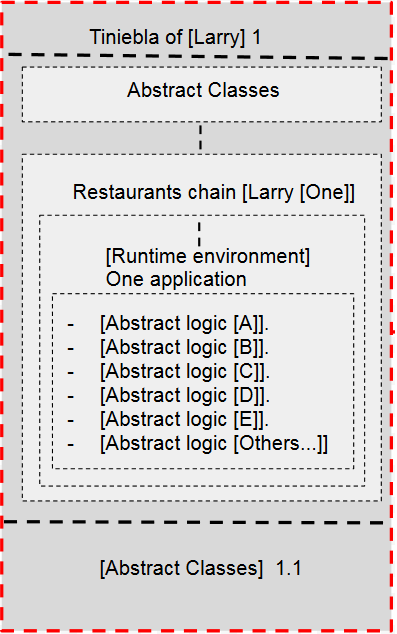
## 

## 

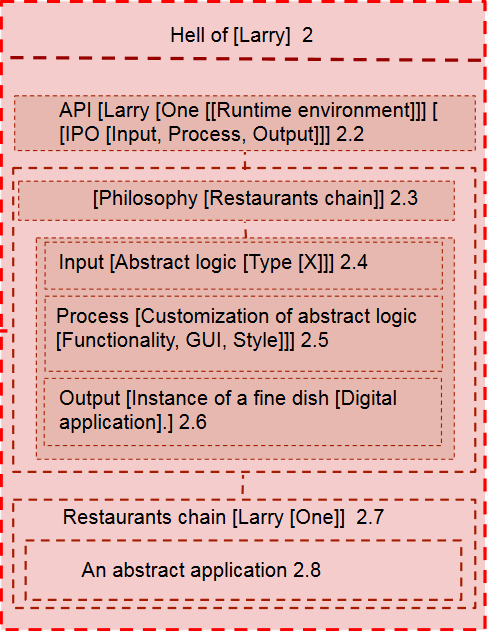
## 

## Abstract Classes:

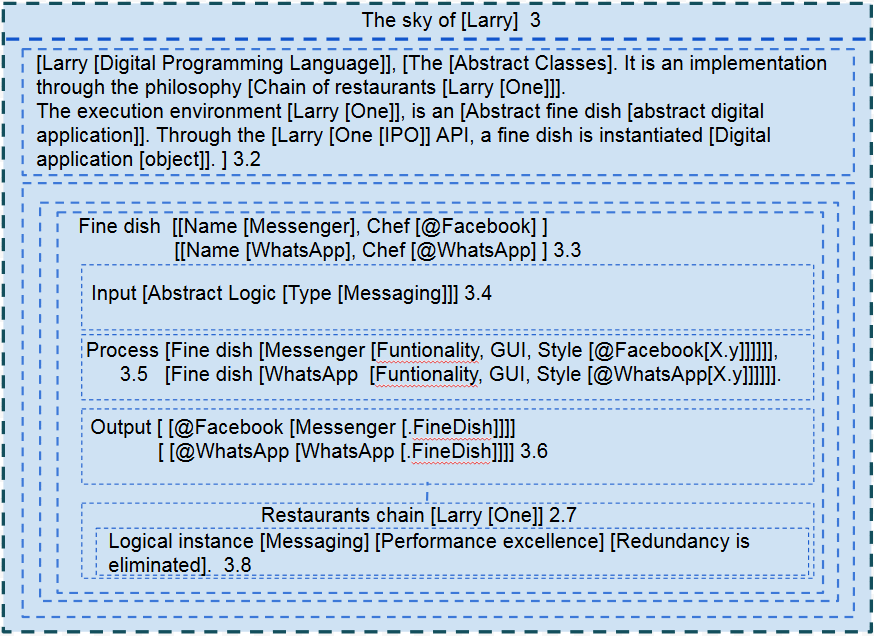
* + [Larry [Digital Programming Language]], [The [Abstract Classes]. It is an implementation through the philosophy [Chain of restaurants [Larry [One]]].
  + The execution environment [Larry [One]], is an [Abstract fine dish [abstract digital application]]. Through the [Larry [One [IPO]] API, a fine dish is instantiated [Digital application [object]]. ].
  + [Abstract logic [Type [X]]].
  + Process: [Customization of abstract logic [Functionality, GUI, Style]]].
  + Output: [Instance of a fine dish [Digital application].].
  + Abstract Classes:



* + An abstract application:



* + Restaurants chain [Larry [One]]



## Abstract Classes:

* + [Conditional] [Recipes] is equal to Interfaces.

## Logo:

