# Architecture of the project

The current document will describe the KitchenHeaven project from an architecture perspective.

Table des matières

[Architecture of the project 1](#_Toc151328413)

[I Physical Architecture 1](#_Toc151328414)

[II SoftWare architecture 1](#_Toc151328415)

[III Risks 1](#_Toc151328416)

## I Physical Architecture



### Front-End

The Web Application is installed on a dedicated serveur which will be accessible from internet.

### BackEnd

The Web API is installed on a dedicated server which haven’t access to internet by default. As he will request an external API (TheMealDbAPI), a server proxy will be set on to provide the access.

### Database

The database will be installed dedicated server. The API server will be the only server allowed to access it.

## II SoftWare architecture

The main purpose of the software architecture is to separate the responsabilities between each layer. Then each one first ignore the way others works, and so it avoid interdependence between parts of the project. It result each layer can elvolve independently.

### KitchenHeaven.Framework (Assembly Back-End)

The framework is the core of the project. It performs all actions needed to apply changes to stored data or retrieve them. The advantage is to have the same object and rules applyied, regardless of which project referenced it.

It need to evolve in parallel of the .net framework to avoid any obsolescence problem

#### DataModel :

The datamodel contains all the object that will be used to transport data beetween each layer of the framework.

#### DataAccess :

The DataAccess layer isolates all operations on data source. It make the interface between data model and stored data. Data can be processed by the business layer regardless to the source.

#### Service :

The Service layer applies the business rules during the pocess of data. It validates all request fom business perspective and organize operations and call data access to perform the operations on data

KitchenHeaven.API (Web API)

The Web API is the the intermediary beetween user requests and the service. This is a restfull API based on the http protocole which means client don ‘t have to manage any specificity to request the API.

Moreover, the API manage all security and request’s content validation. Which stop invalid process earlier.

### Presentation

The Presentation Layer provides an interface for user to perfom their request more friendly. Using a javascript framework to make the web application make it more portable. And with the Single Page Application architecture, it reduce the load time of each page.

As the processed data are based on a json, graphic interface and user interaction are perfom independently of the service implementation.

## III Risks

#### System - HttpRequest Security

The main purpose of the API is to be a bridge between a client application and the core of the data storage. In the first version, only the web application will request the API, it need no more than configure a firewall to block any request expect from the web server.

This could change in a further version, notably in case of a mobile application with wich « all internet » could have to access to the API server.

A reverse proxy could then be neccessary for security purpose. First the reverse-proxy will be the server known outside the network. In the other hand, all request to the API will be check in accordance with security rules.

#### System - trafic extension

The current version of KitchenHeaven won’t be heavy in the first months or years. But the more users or use there will be, the more request will be sent in each layer.

In a physical level, adding servers with load balancing can be an answer. Programatically, it could also be a good improvment to implement a limit of rate request to proceed.

Application – API Data Access

To request TheMealDbAPI, four classes (MealCategoryAPIAccess, MealAreaAPIAccess, MealIngredientAPIAccess, MealAPIAccess) implement the same interface IMealAPIAccess.

This avoir the dependency injection and a simple use of unit test for all methods which use the API.

Four specific interface IMealCategoryAPIAccess, IMealAreaAPIAccess, IMealIngredientAPIAccess and IMealAPIAccess, implementing IMealAPIAccess, should be created and used in the constructor of IMenuService.

Application – External API usage

Depending on an external API can be dangerous for KitchenHeaven. If the API is unavailable or doesn’t exist anymore, the entire project can be useless. The first step to avoid this was to add Restaurent’s meal in the database, so user scan consult them at anytime.

The problem is the desynchronization between data available in TheMealDB and saved meal in database. To resolve this, the solution can be study to permit administrator of KitchenHeaven to update existing data in database with TheMealDB data.

Another way is to study the possibility of implementing an offline application wich will work with local data and synchronize itself with the server later.

#### Application – exposition schema database dans model

At the moment, data are exchanged regardless the size of the messages.

For exemple, sending all the meal’s data to the server when adding a meal to a menu is accepted (). Same, when a meal is searched by his name, the server include all the meal details, which are not used in the meal’s preview.

The two risk are :

* Exchanged data schema is to close to the database schema
* Message are unnecessary

To avoid those problems, i twill be necessary to add in smaller object in data model:

* MealPreviewModel
  + Name
  + Miniature
  + Area
  + Category
* MenuModel
  + List<MealModel>

Application – Search Restaurant Result

In the first version, result of a restaurant research are displayed in one list. The overall loading time will increase with the number of restaurant. Moreover the result won’t be clear and workable for the users.

Two ways to correct this behaviour are :

* Implement paging in the result
* Implement filter on the result

Application – restaurant chain

As two restaurant can’t have the same businessIdentifier, Restaurant chain are not managed now.

There are two improvment for the application regarding this point :

* Adding a column « ParentRestaurantId » in the restaurant table to create a link beetween restaurant and make the chain
* Exporting adress in another table

The first one will allow to manage restaurant chain and their franchise restaurant. The second one limit the data storage size, as menu, restaurant chain informations are the same for all restaurant of a chain. But franchise restaurant should be created independently

#### Application Acceptation - wait from api return

In the presentation layer, all actions wait the return of the API request to complete their process. Users wait on the screen during the execution time.

API request can be made asynchronously, the user can work on KitchenHeaven and a toaster component alert him when one of his action is finished. So he has the choice to use the application as he wants

#### Application Acceptation – Add meal to menu

For this version, add a meal to menu is quite tedious :

* Restaurant must be created and/or selected
* Meal are added one by one

Making the search meal accessible from the welcome page and adding a button « selects restaurant » will prevent the first default. For the second, added meal can be added to a « basket » and edded to menu at once. Moreover, a checkbox can be added to « search meal » result and the « Add to menu button » add all selected meal to the menu.