NutriStyle

12/03/17

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision History** | | | |
| **Version** | **Date** | **Description** | **Author** |
| 1.0 | 12/03/17 | Initial documentation | Caleb Skinner |
| 1.1 | 17/03/17 | Added JavaScript example | Caleb Skinner |
|  |  |  |  |
|  |  |  |  |

logo.gif

|  |  |
| --- | --- |
| **Definitions** | |
| Plugin | Custom code for Microsoft CRM. Runs synchronously. Can be configured to run before the entity is created, after the entity is created, on delete, merge, etc. |
| Workflow | Custom code for Microsoft CRM. Runs asynchronously. Created by the Microsoft CRM admin via the workflow interface. Can have custom code added to it if needed. |
| SSRS | Microsoft SQL Server Report Server. |
| CRM | Microsoft Customer Relationship Management. |
| SQL | Microsoft SQL Server. |
|  |  |
|  |  |

**The NutriStyle technical architecture**

The solution leverages the power of Microsoft CRM has a development platform. The database layer is Microsoft SQL Server and reports are handled with SSRS. Licensing is not an issue as this whole model can run with the ‘external connector license’ with one CRM user.

The application uses a typical three tier development approach:

1. Visual presentation
2. Managing the requests from the visual presentation and interfacing with the database
3. Database

The existing visual presentation is written in Silverlight. All the data and metadata required to make the visual layer function are pulled from Microsoft CRM with a number of web service calls. As data flows from the visual layer into the management layer various plugins are fired to execute custom code against the various entities. The plugin code can be found in Applications\Plugins. After the user enters in their various choices, weight, gender, height, meal plan, food likes, etc the Menu Generator is fired and generates a menu. See Applications\MenuGenerator for the code. The Menu Generator creates a menu using an innovative process of food management, substitution portion size, etc to create a menu that matches what the user is looking for.

There are existing reports to show signups, shopping lists, food logs, etc. Reporting for internal use can take advantage of the wizards to assist in reporting.

The multitenancy of Microsoft CRM means that each grocer or NutriStyle partner can have their own database (foods), while retaining the all the data management abilities of the reports, plugins, workflows, middle tier, menu generator, etc. The plugins and workflows can be configured to run per instance. So instance ‘A’ can have different data management processes than instance ‘B.’ This configuration is wizard driven and quite simple to perform.

Microsoft CRM and also be ran out of the cloud. So scaling the application becomes much simpler task. The menu generator code can stay in a NutriStyle controlled environment as it is running a web service.

The current architecture can be leveraged to quickly produce a HTML front end. All the web service calls are in place, the middle tier is functionable and the database layer is working. The existing Silverlight layer can be modified to create a Universal Windows 10 application. Mobile platforms (cell, tablet, etc) can also use the existing model.

The code is all written in C#. It can be quickly ported to Mono if desired.

The existing webservices can be seen here: <http://dynamiconnections.com/Nutristyle/Webservices/Pages/WebServices.asmx> - Live link. They can be consumed by anything that can create a SOAP envelop.

Here’s a quick example of connecting to a very similar web service in JavaScript:

function login() {

var userName = document.getElementById("userName").value;

    var password = document.getElementById("password").value;

request = null;

request = getRequestObject();

   var address = "/Pages/Webservices.asmx/Login";

    var data = "userName="+escape(userName)+"&password="+escape(password);

    sendRequestWithData(address, data, loginResponse, request);

}

function loginResponse() {

if (request.readyState == 4) {

if (request.status == 200) { //Successful Request

var xmlDoc = getXmlDoc(request.responseText);

var error = xmlDoc.getElementsByTagName("error")[0];

            if(error != null) {alert("Login AJAX call failed: "+error.text);}

            else

{

             //Perform custom stuff here

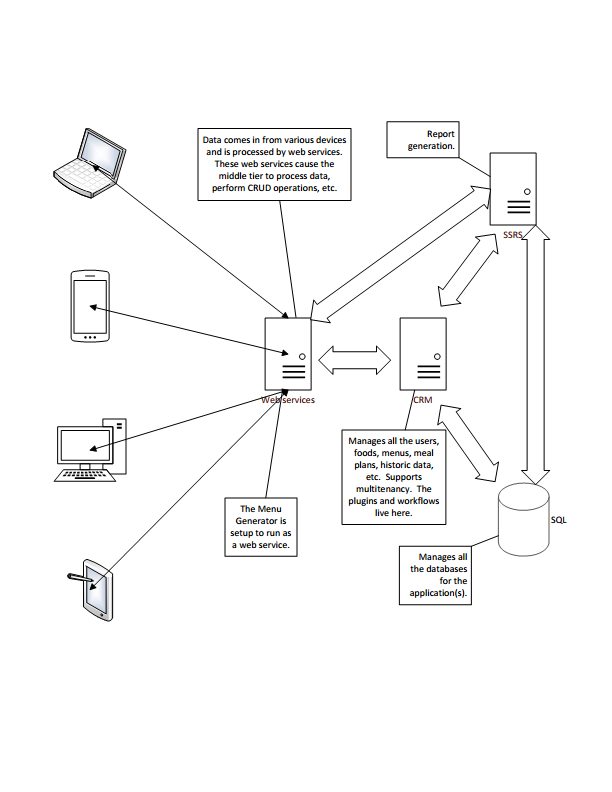
            }

        }else {alert("Error AJAX Call to Login failed. "+request.responseText);}

}

}

sendRequestWithData is a function that can be found in the attached ajax.js

**agc.gif**

#### Dynamic Connections

Caleb Skinner

Partner

[caleb@dynamiconnections.com](mailto:caleb@dynamiconnections.com)

Caleb. Skinner @ skype