Интерфейс для обмена данными между фронт-офисным ресторанным модулем ТРАКТІРЪ и любым ВЭБ-сайтом или .NET-приложением (реализация электронного меню в ресторане)

v1.0.0.4

Назначение:

Интерфейс разработан с целью обеспечения взаимодействия фронт-офисного модуля ТРАКТІРЪ и Электронного Меню для посетителей заведения общественного питания (внешней вэб-страницы или .NET приложения, работающих по протоколу http)
Интерфейс обеспечивает передачу заказа в обработку непосредственно с вэб-страницы (автоматическое создание нового заказа на фронт-офисном модуле и распечатка «встречек» в соответствии с признаком локализации: бар и три кухни) а так же получение вэб-страницей по запросу актуального справочника меню с сервера ресторанного модуля.

Как работает:

Для обеспечения работы интерфейса необходимо наличие двух модулей программы «RestoMarket»: основной фронт-офисный модуль для ресторанов, кафе - модуль «ТРАКТІРЪ» и серверная часть модуля «RRC Waiter» (мобильный официант)

Обмен данными между Вашей вэб-страницей или .NET приложением, на которых реализовано Электронное меню происходит с помощью обыкновенных GET или POST-запросов. Запросы направляются на локальный http-сервер, на котором работает серверная часть модуля Мобильный официант. Сервер должен поддерживать php: оптимальные версии 5.2 — 5.6

Ваша вэб-страница или .NET приложение смогут самостоятельно с помощью данного интерфейса создавать заказы или дозаказы, и получать обновления справочников.

Электронные меню, разрабатываемые под конкретных клиентов компанией Restaurant Retail Control представляют из себя сайты на wordpress, но только эти сайты работают внутри Android-приложения (как внутри оболочки: через наш собственный браузер). Таким образом Электронные меню представляет из себя полноэкранное приложение, в котором нет лишних элементов, посетитель даже не видит адресную строку, а также есть возможность обмена данных между страницей сайта с Android-приложением и сохранения в нём настроек.

Описание запросов:

/kontr.php **контрольный запрос**

GET-запрос предназначен для контроля связи с сервером обработки заказов.
Он отправляет на фронт-офис команду, результатом которой является распечатка на всех подключенных к фронт-офисному модулю принтерах небольших «контрольных встречек»
пример: `http://192.168.0.2:80/Menu/kontr.php`
запрос не имеет параметров и сервер ничего не возвращает, кроме распечатки «контрольных встречек»

/visitormenu.php **справочник меню**

запрос предназначен для получения от сервера всего справочника меню и его можно использовать для периодического обновления наименований и текущих цен на сервере Вашей вэб-страницы, в случае изменения цен или наименования.
пример: `http://192.168.0.2:80/Menu/visitormenu.php`
Запрос можно передавать в двух вариантах, как GET-запрос (в этом случае сервер возвращает полный справочник меню формате xml) или как POST- запрос с параметрами прямого ODBC подключения к MySQL-базе данных Вашей вэб-страницы (в этом случае сервер, используя доступ к базе данных, автоматически обновит наименования и цены меню Вашей вэб-страницы)

Вариант 1: Если запрос передаётся Вами как **GET-запрос**, то он не имеет параметров,
сервер в ответ на GET-запрос просто возвращает полный справочник меню в формате xml-строки:

```
<?xml version="1.0" encoding="utf-8"?>
<MENU>
	<item cod="304" name="AZNAURI Коньяк 3* 0,25 л" price="72.10"/>
	<item cod="305" name="AZNAURI Коньяк 3* 0,5 л" price="135.10"/>
	<item cod="306" name="AZNAURI Коньяк 4* 0,5 л" price="148.70"/>
	<item cod="307" name="AZNAURI Коньяк 5* 0,5 л" price="157.70"/>
	<item cod="308" name="AZNAURI Коньяк 5* 0,25 л" price="81.10"/>
	...
	...
	...
	<item cod="183" name="AZNAURI Вино 0,75 п.сладкое" price="89.40"/>
</MENU>
```

где: cod, name, price соответственно код, наименование и розничная цена товара

Вариант 2: Если запрос передаётся Вами как **POST-запрос**, то в теле запроса необходимо передать строку с параметрами прямого ODBC подключения к MySQL-базе данных Вашей вэб-страницы:

```
host=your_mysql_host&user=your_login&pass=your_password&base=database_name&port=3306&cpage=cp2151
&table=table_name&cod=cod_colum_name&name=name_colum_name&price=price_colum_name
```

где: host - имя хоста (или ip адрес) MySQL-сервера, где находится база данных вэб-страницы
user - логин (должны быть открыты права на доступ на UPDATE таблицы с товарами)
pass - пароль
base - имя базы данных
port - порт сервера (не обязательно: если параметр не передавать, то по умолчанию port = 3306)
cpage - кодировка таблицы с товарами в формате php, например cp1251 или utf-8 (не обязательно: если параметр не передавать, то по умолчанию cpage = utf-8)
table - имя таблицы с товарами Вашей вэб-страницы
cod - имя колонки в таблице с товарами, где хранится Код товара
name - имя колонки в таблице с товарами, где хранится Наименование товара
price - имя колонки в таблице с товарами, где хранится Цена товара

POST-запрос автоматически обновит наименования и цены уже существующих товаров в указанной Вами таблице sql-запросами: `UPDATE table SET name = 'name', price = 'price' WHERE cod = 'cod'`
и в ответ вернёт Вам xml-строку:

```
<?xml version="1.0" encoding="utf-8"?>
<RESULT send="457"/>
```

где: send - количество отправленных сервером sql-запросов

/visitorinsert.php **отправить Заказ на сервер**

этим GET-запросом передаётся массив данных, содержащий заказ клиента с Вашей вэб-страницы.
например:

```
http://192.168.0.2:80/Menu/visitorinsert.php?data=1285*1000*0*|1391*2000*0*посолить и поперчить|
3192*10000*0*засунуть куда-нибудь|1400*1000*0*&ofc=7&stl=1&ter=0&dop=0
```

в запрос необходимо передать параметры:

data (массив строк) - каждая подстрока массива соответствует одной строке заказа клиента и содержит четыре элемента: код товара
количество х 1000
разделитель (в данном контексте не используется) необходимо передать 0
текстовое сообщение повару - да, да, можно передать сообщение для повара и это сообщение будет распечатано на встречке!

элементы подстроки разделены между собой символом *, а подстроки массива разделяются символом | .

ofc (число) - код официанта, который обслуживает этот заказ (из справочника официантов)
stl (число) - номер стола
ter (число) - если новый заказ передать 0, если дозаказ: номер терминала существующего заказа
dop (число) - если новый заказ передать 0, если дозаказ: номер существующего заказа

в нашем случае сервер возвращает такую xml-строку:

```
<?xml version="1.0" encoding="utf-8"?>
<RESULT send="4"/>
```

где: send - количество принятых сервером строк массива, если оно отличается от количества строк в отправленном запросе, то send=0

Примечание: этот запрос является исполнительной командой для сервера создать заказ в базе данных и распечатать встречки! Отменить его, как и другие заказы, можно только карточкой суперадминистратора

/visitorkassa.php **открытые заказы**

GET-запрос предназначен для получения от сервера списка открытых заказов для создания дозаказа
пример: `http://192.168.0.2:80/Menu/visitorkassa.php?ofc=7&stl=13`
в запрос могут быть переданы параметры в качестве фильтра (если параметра нет - то по всем):

ofc (число) - код официанта
stl (число) - номер стола

сервер возвращает список открытых заказов в формате xml-строки, отфильтрованный в соответствии с отправленными Вами в запросе параметрами:

```
<?xml version="1.0" encoding="utf-8"?>
<KASSA>
  <item ter="0" num="1242" sum="1067.50" stl="7" ofc="13" stat="Заказ" dat="2019-02-23 18:40:55"/>
  <item ter="1" num="145" sum="463.80" stl="7" ofc="13" stat="Дозаказ" dat="2019-02-23 19:57:13"/>
</KASSA>
```

где: ter, num, sum, stl, ofc, stat, dat соответственно номер терминала, номер заказа, сумма заказа, номер стола, код официанта, статус и время заказа

/visitorpers.php **справочник официантов**

GET-запрос предназначен для получения от сервера справочника официантов и его можно использовать для идентификации входа официантов на Вашу вэб-страницу.
пример: `http://192.168.0.2:80/Menu/visitorpers.php`
запрос не имеет параметров
сервер возвращает справочник официантов в виде xml-строки:

```
<?xml version="1.0" encoding="utf-8"?>
<PERSON>
      <item cod="4" name="Анастасия" pass="567"/>
      <item cod="5" name="Евгения" pass="788"/>
      <item cod="3" name="Александра" pass="134"/>
</PERSON>
```

где: cod, name, pass соответственно код, имя официанта и числовой пароль для входа в программу

Примеры использования:

```
<script type="text/javascript">

     var server = 'http://192.168.0.2:80/Menu';

     function getXmlHttp(){                         // кроссбраузерная функция создания объекта XMLHttpRequest
          var xmlhttp;
          try {
                 xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
          } catch (e) {
                 try {
                      xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
                 } catch (e) {
                      xmlhttp = false;
                 }
          }
          if (!xmlhttp && typeof XMLHttpRequest !== 'undefined') {
                 xmlhttp = new XMLHttpRequest();
          }
          return xmlhttp;
     }

                        // Функция передачи POST-запроса на сервер для автоматического обновления меню на сайте
     function updateMenu(host, user, pass, base, port, cpage, table, cod, name, price){

           var params = 'host=' + encodeURIComponent(host) + '&user=' + encodeURIComponent(user);
           params = params + '&pass=' + encodeURIComponent(pass) + '&base=' + encodeURIComponent(base);
           params = params + '&port=' + encodeURIComponent(port) + '&cpage=' + encodeURIComponent(cpage);
           params = params + '&table=' + encodeURIComponent(table) + '&cod=' + encodeURIComponent(cod);
           params = params + '&name=' + encodeURIComponent(name) + '&price=' + encodeURIComponent(price);

                var xmlhttp = getXmlHttp();
                xmlhttp.open('POST', server+'/visitormenu.php', true);           //асинхронный POST-запрос
                xmlhttp.setRequestHeader('Content-Type', 'application/x-www-form-urlencoded');
                xmlhttp.onreadystatechange = function() {         //событие при изменении состояния запроса
                        var parser, xmlDoc, send;
                        if (this.readyState === 4 && this.status === 200) {
                             parser = new DOMParser();                              //разбираем xml-строку
                             xmlDoc = parser.parseFromString(this.responseText,"text/xml");
                             send = xmlDoc.getElementsByTagName('RESULT')[0].getAttribute('send');
                             alert('Update ' + send + ' records completed successfully');
                        }
                };
                xmlhttp.send(params);
     }
                                                 // Функция передачи заказа клиента на сервер для обработки
     function sendZakaz(data, ofc, stol, ter, dop){
          // формат строки с данными (data) '1285*1000*0*|1391*2000*0*посолить и поперчить|3192*10000*0*'

          var params = 'data=' + data + '&ofc=' + ofc + '&stl=' + stol + '&ter=' + ter + '&dop=' + dop;

                var xmlhttp = getXmlHttp();
                xmlhttp.open('GET', server+'/visitorinsert.php?'+params, true);   //асинхронный GET-запрос
                xmlhttp.onreadystatechange = function() {         //событие при изменении состояния запроса
                        var parser, xmlDoc, send, record;
                        if (this.readyState === 4 && this.status === 200) {
                             parser = new DOMParser();                              //разбираем xml-строку
                             xmlDoc = parser.parseFromString(this.responseText,"text/xml");
                             send = xmlDoc.getElementsByTagName('RESULT')[0].getAttribute('send');
                             record = xmlDoc.getElementsByTagName('RESULT')[0].getAttribute('record');
                             if ( send > 0 && record === send ) {
                                    alert('Send ' + record + ' records completed successfully');
                             }
                        }
                };
                xmlhttp.send(null);
     }

</script>
```

```
<script type="text/javascript">

	var server = 'http://192.168.0.2:80/Menu';
	var zakazList = [];          // массив «список открытых заказов» состоит из экземпляров класса Zakaz()

                                     // конструктор класса Zakaz(), содержит строку с параметрами заказа
                                                                  // является элементом массива zakazList[]
	function Zakaz(ter, num, sum, stl, ofc, stat, dat){
	    this.ter = ter;
	    this.num = num;
	    this.stl = stl;
	    this.ofc = ofc;
	    this.sum = sum;
	    this.title = stat + ' ' + dat.toString();
	}

                                // Функция загрузки списка открытых заказов с сервера в массив zakazList[]
	function getKassa(stl, ofc){
	         var params = 'stl=' + stl + '&ofc=' + ofc;
	        var xmlhttp = getXmlHttp();
	        xmlhttp.open('GET', server+'/visitorkassa.php?'+params, true);   // асинхронный GET-запрос
	        xmlhttp.onreadystatechange = function() {         // событие при изменении состояния запроса
	                 var parser, xmlDoc, x, i, ter, num, sum, stl, ofc, stat, dat, zakaz;
	                if (this.readyState === 4 && this.status === 200) {
	                     zakazList.length = 0;                                  // очищаем массив
	                     parser = new DOMParser();                              // разбираем xml-строку
	                     xmlDoc = parser.parseFromString(this.responseText,"text/xml");
	                     x = xmlDoc.getElementsByTagName('item');
	                     for (i = 0; i < x.length; i++) {
	                            ter = x[i].getAttribute('ter');
	                            num = x[i].getAttribute('num');
	                            sum = x[i].getAttribute('sum');
	                            stl = x[i].getAttribute('stl');
	                            ofc = x[i].getAttribute('ofc');
	                            stat = x[i].getAttribute('stat');
	                            dat = x[i].getAttribute('dat');
	                            zakaz = new Zakaz (ter, num, sum, stl, ofc, stat, dat);
	                            zakazList.push(zakaz);     // добавляем экземпляр класса Zakaz() в массив
	                     }
	                }
	        };
	        xmlhttp.send(null);
	}

</script>
```