1.- How long did you spend on the coding test?

It took about 4 hours to do it. I have tried to do it as fast as I can, but I like developing in both, C# and JavaScript, I spent a little more time in creating a web api to get the list of restaurants in C# and then displaying them using BackboneJS. Moreover, it took me a little bit longer than the 3 expected hours, because I had some problems mocking the HttpClient object, to get fake responses from the api and not do real calls to it.

What would you add to your solution if you had more time? If you didn't spend much time on the coding test then use this as an opportunity to explain what you would add.

I would probably spend a little more time in error handling, in case the server fails. Moreover, I would spend more time in the css and how the results are shown. Right now it is not quite attractive.

2.- What was the most useful feature that was added to the latest version of your chosen language? Please include a snippet of code that shows how you've used it.

In C#, in .Net 4.5 they had included HttpClient to easily deal with Get and Post requests. It is not something really really new, but it is quite nice and makes developers life a little bit easier. I have used it to get the list of restaurants from the justEAT api:

```
private async Task<IList<Restaurant>> GetRestaurantFromAPI(string outcode)
{
	JustEATResponse result = new JustEATResponse();
	_httpClient.BaseAddress = new Uri(ConfigurationManager.AppSettings["JustEATUrl"]);
	AddHttpClientRequestHeaders();
	HttpResponseMessage response = await _httpClient.GetAsync(string.Format("{ 0}?
q={ 1}",
		ConfigurationManager.AppSettings["RestaurantEndPoint"], outcode));
	if (response.IsSuccessStatusCode)
	{
		result = await response.Content.ReadAsAsync<JustEATResponse>();
	}
	return result.Restaurants;
}
```

And it is really easy to set the required headers:

```
private void AddHttpClientRequestHeaders()
{
	_httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue(
			ConfigurationManager.AppSettings["AuthorizationScheme"],
			ConfigurationManager.AppSettings["AuthorizationParameter"]);
	_httpClient.DefaultRequestHeaders.AcceptLanguage.Clear();
	_httpClient.DefaultRequestHeaders.AcceptLanguage.Add(new
StringWithQualityHeaderValue(
			ConfigurationManager.AppSettings["AcceptLanguage"]));
	_httpClient.DefaultRequestHeaders.Host = ConfigurationManager.AppSettings["Host"];
	_httpClient.DefaultRequestHeaders.Add("Accept-Tenant",
			ConfigurationManager.AppSettings["AcceptTenant"]);
}
```

Also, using HttpClient, make testing easier, I just needed to create a fake HttpMessageHandler and mock the responses sent:

```
Mock<FakeHttpMessageHandler> msgHandler = new Mock<FakeHttpMessageHandler>() { CallBase =
true };
		msgHandler.Setup(t => t.Send(It.Is<HttpRequestMessage>(
			msg =>
			msg.Method == HttpMethod.Get &&
			msg.RequestUri.ToString().Contains(string.Format("restaurants?
q={ 0}", outcode)))))
```

```
                    .Returns(response);
            return msgHandler;

…

HttpClient fakeHttpClient = new HttpClient(msgHandler.Object);
```

In the JavaScript part, in BackboneJs, I've been reviewing the changelog, and the latest versions the mainly fix bugs and stuff. I can emphasise the good parts of using it, like it really separate the MV* duties, with clear views, models and collections. It is really easy to manage the changes in the models and collections listen to the events they trigger.

```
me.listenTo(me.Restaurants, 'reset', me.addRestaurants); // in app-view.js
```

And no need to do anything to speak with the server (web api), because it provides the url property for the end point and the fetch, save, get, …, functions.

```
app.RestaurantCollection = Backbone.Collection.extend(/** @lends
RestaurantCollection.prototype */ {
      /**
     * @property {Object} url Server route to get collection
     */
      url: '/api/restaurant',
      /**
     * @property {Object} model Associates RestaurantCollection with a specific model
(RestaurantModel)
     */
      model: app.RestaurantModel
});


me.Restaurants.fetch({ data: $.param({ outcode: outcode}), reset: true }); // it will
make a get request http://localhost:8080/restaurant?outcode=<outcode>
```

3.- How would you track down a performance issue in production?

I would check first how the server is behaving, how much memory is using, the number of files opened, and I would look for any kind of exceptions in the log. That last thing would give me a clue of what the server is doing and if any of the resources is failing.

If it is not related to the server, I would take a look to the data base and check if there is any query or transaction that it is taking longer to execute, so it is making everything slower.

Moreover, I would take a look to the Services running in case one of them is down, and this is the cause of all the errors.

Have you ever had to do this?

Yes, when I worked as a system administrator I had to track performance issues in the Linux machines where the application where running. Usually the main problems were stuck processes trying to get any kind of resource, file access or database access, and out of memory problems.

Also, in my actual work, we have to deal with life problems in our Windows Server machine where the application run, taking care of the performance of the IIS, and all the services that are running un the background, processing the data in the data base and taking care of all the reading streams.

4.- How would you improve the JUST EAT APIs that you just used?

I would add some more doc, for example when opening in the browser http://api-interview.just-eat.com/, it can display all the possible calls and some more info on the returning objects.

Also, I would add more parameters to the restaurants call, to retrieve them in chunks in case we do not want to get the whole list at once. Maybe modified the returned object, and only return the list of restaurants, the outcode is already known, because the user is setting it.

Furthermore, I would add a more complex authentication method, such us Oauth or something similar, based in some kind of token exchange.

5.- Please describe yourself using JSON.

```
{
	name: 'Maria',
	surname: 'Jimenez Campos',
	gender: 'female',
	birthDate: '04/10/1983',
	birthCountry: 'Spain',
	birthCity: 'Jaen',
	livingCountry: 'United Kingdom',
	livingCity: 'London',
	jobs: [
		{
			company: 'Onalytica',
			jobTitle: 'Software Developer',
			actualJob: true
		},
		{
			company: 'Unit4 R&D',
			jobTitle: 'Software Developer',
			actualJob: false
		},
		{
			company: 'BMN',
			jobTitle: 'System Administrator',
			actualJob: false
		},
		{
			company: 'Caja Granada',
			jobTitle: 'System Administrator',
			actualJob: false
		}
	],
	education: [
		{
			school: 'University of Granada',
			city: 'Granada',
			country: 'Spain',
			name: 'Master of Science in Software Development'
		},
		{
			school: 'University of Granada',
			city: 'Granada',
			country: 'Spain',
			name: 'Postgraduate Certificate in Education at University of Granada'
		},
```

```
            {
                school: 'University of Granada',
                city: 'Granada',
                country: 'Spain',
                name: 'Degree in Computer Science'
            },
            {
                school: 'Sta Maria de la Capilla',
                city: 'Jaen',
                country: 'Spain',
                name: 'High School'
            }
        ],
        techSpecs: {
            programmingLanguages: [
                'C#',
                'Java',
                'JavaScript',
                'C++',
                'ShellScript',
                'Perl',
                'Python'
            ],
            operatingSystems: [
                'Windows',
                'Linux'
            ],
            frameworks: [
                'ExtJS',
                'BackboneJs',
                '.Net',
                '.Net MVC',
                'EntityFramework',
                'Jquery',
                'AngularJS',
                'Ionic'
            ]
        },
        hobbies: [
            'running', 'swimming', 'knitting'
        ]
}
```