# Developer Survey and Coding Exercise

## Technical Expertise

What is your level of experience with the following technologies and frameworks?

Please answer each question with **1** for "no experience", **2** for "some experience" or **3** for "considerable experience".

### Languages

* C#: \_1\_\_
* Javascript: \_3\_\_
* Typescript: \_3\_\_
* HTML/SCSS: \_3\_\_

### Frameworks and Technologies

* .NET: \_1\_\_
* .NET Core: \_1\_\_
* Angular: \_3\_\_
* NgRx: \_1\_\_
* GraphQL: \_1\_\_

## Experience and Training

Please list any computer-related training, qualifications and work experience you have.

* First job: Used javascript framework “SenchaJS” to create dynamic web apps using IBM Lotus Domino as a server. Also worked with Web Services to exchange data in XML. Worked with Java in Lotus Domino to communicate with an MQ Server solution.
* Second job: Worked with Pearl for web pages hosted on IBM IHS and managed middleware in AIX. Worked with Angular and Nodejs to create web apps hosted on an OpenShift cloud, using Cloudant as database hosted in the IBM Cloud, I worked in the architecture design and the implementation.
* I just have a certification as developer in Lotus Notes technologies, from 2013.
* In the school (Bachelor and High school) I worked a lot with C++, Visual Basic and Java.

## Exercise

The goal is to create a simple server-client "To-Do" application using .NET Core as the server and Angular as the client. The data sent between server and client will be in JSON format. In the ZIP file you received you will find the base source code with instructions on how to compile and run it.

For now, we have only created a basic server controller returning a list of to-do items and a simple client app which shows the list to the user. To-do items must be formatted in title case (the first letter of every word should be capitalized).

Feel free to add any comments that could be useful for the reviewer. In addition, unless otherwise indicated, you can use any library of your choice.

Please reply to the questions of this exercise directly in this document to submit it back for review.

### Code Review

Review the existing code in the following files:

#### Server

* **Controllers/TodosController.cs**
  + Take note of the ToTitleCase function and its unit tests.
* **Controllers/TodosControllerTests.cs**

#### Client

* **todo-list/todo-list.component.ts**

Please, have the following points in mind:

* Standards, good practices you know of
* Code re-usability (for example, we may need to use ToTitleCase in other places too)
* The input data could be very large (your program could be run on a machine with limited memory)

**Please list all the things you would change or improve in the code specifying the line number whenever necessary and explain the reasons.**

* **Controllers/TodosController.cs**
  + **GetList method**
    - To avoid memory issues, use ReadLines method from File class, for each will iterate each line and get one by one until the last line is reached and nothing is returned.
    - Improve custom ToTitleCase: move the method to a model to be used within the ToDo data structure, also when the string contains more than one space between words or start with some spaces or ends with spaces the actual method will fail, I change the order of the recursive method to first check if there are spaces to split and the check if that space is the first character. Add a condition for 2 consecutive spaces and return both spaces to concatenate in the final string and then continue with the recursive method.
    - Other thing could be: Implement Globalization libraries and use the TextInfo.ToTitleCase method.
* **Controllers/TodosControllerTests.cs**
  + Add the following cases:
    - When string is multiple character
    - When string starts or ends with spaces
    - When string contains consecutive spaces between words
* **todo-list/todo-list.component.ts**
  + To improve I should split the code in different files:
    - Html file to load all the User interface instead to put all in the “template” property of the component.
    - SCSS file to apply some custom styling for this specific component, all the global styles will be loaded from the styles.scss file int the root of angular project.
    - Service file to implement all the calls to the ToDo’s API, in this example to implement the get todos http call.
  + When all the service, scss and html files were created, the changes on the code should be:
    - Move lines from 7 to 15 to the Html file. “template” property should be empty.
    - “styles” property on line 17 should be the name of the SCSS file instead of the inline styles.
    - Should remove http get call from line 23 and implement in NgOnInit method, this will load the data once the component is initialized.
    - The http get call should be implemented in a Service file.

**What possible enhancements would you consider?**

* Use SCSS instead of CSS files, this will be helpful to use dynamic styles. Give us an opportunity to create conditional styling.

### Refactor

Refactor the code in Controllers/TodosController.cs and todo-list/todo-list.component.ts applying the changes and improvements you suggested in the code review.

### Add Create and Delete To-Do Items

Add the ability to create and delete to-do items. User input and stored data should always be valid and safe. You may store data in a text file or in a in-memory database.

**What kind of storage would you use in a real application to store the to-do items? Explain why.**

* Since is a web application, I should use a cloud database, for example CouchDB that is a non-relational database that is easy to use.

### Add New Fields

Add a couple of new optional fields to the to-do items: due date and notes.

Create a new detail view to edit a single to-do item. From the current list, the user should be able to click one and navigate to the new view containing:

* Title
* Indication of completion
* Due date
* Notes

The user should be able to edit the fields and save or cancel the changes.

Feel free to enhance the user interface.

### Add a Store (Optional)

**NOTE by Israel Salazar:** I didn’t implement this part because I never used NgRx before, but I take some time to review and learn the basics. I found this very useful because we could isolate all the interactions between the Front end application and the API calls, I’ve always worked with a service based apps, but I found kind of difficult implement automated testing with this basis because the results depends always on the server data. Nevertheless, implementing an NgRx Store solve that problem giving to the components an observables objects to interact and we could create specific output for each state and don’t use trigger the effects when we are testing. I liked this technology and I think I could use it in the future when I understand it better.

Using NgRx, implement a store in the client app.

**What are benefits of using a centralized store? Explain.**

**What factors would you consider when designing the data structure? Explain.**

### Additional Questions

**How would you implement translations?**

* For the front end I will use json files for each language, on each file I will store the messages and other stuff to show to the user and then load this via service. The service receives the language as parameter and load messages as needed.
* For the server side I could use the internationalization features of C#, I’m not an expert but reviewing some documentation there are some C# features that implements a solution for this. Uses a kind of tables to store each message using a key and we need to create one for each language we need.

**What problems would providing internationalization and localization for many languages have?**

* The encoding is a first problem to solve since some languages have special characters and other format rules that will affect showing data on the page and also receive any input for the database like the description of the todo.
* The dates are also culture affected but the solution implemented here is to convert the data objects to milliseconds (UTC Format) and to show it to the user we could use the system specific format.

**How would you ensure the accessibility of the app?**

* The framework I used (angular material) have some accessibility features that I could implement to display the colors or make readable the content by a screen reader. I just need to implement those features in the app.

**What types of tests would you implement to ensure QA?**

* Unit testing will be useful. Also, we could try automation testing using any technology like Selenium, also the Karma framework implemented on angular projects is an option.
* Also, a User Acceptance Testing could give us a useful feedback before a move to production, we could catch some errors that the automation could not find.

**How would you implement security in your application? Explain.**

* First step will be enable https with a trusted certificate, enable also encryption in the data sent to the APIs instead of use plain text on each call.

## Final Steps

* Zip up the source files (excluding the node\_modules, binaries and build folders)
* E-mail it back to us with this document filled out