Blue Orange
Ronda de Sant Pere 3, Pl. 1
08010 Barcelona, Spain.
http://weareblueorange.com

Back-end Code Challenge

Last updated: June 2017

OVERVIEW

Welcome to Blue Orange's Back-end Code Challenge. This document contains the specifications for building a small application in order to test your coding and problem-solving skills. You will be given an entire week to deliver a fully functional application according to the requirements provided. You retain full rights on the code delivered, so feel free to upload it to your public repository and iterate on it in the future to showcase your coding skills to the community while building something great that you feel proud of.

PROJECT DESCRIPTION

The project you are about to build is a small **Marvel Super Heroes app**. It will require building an web service API as well as a command line tool for accessing the API. The project is small application that allows users to fetch basic information on and add notes about the creators of the vast Marvel Super Heroes universe. You will want to sign up for a developer account in order to take advantage of their public API. You can do so at https://developer.marvel.com.

There are two applications expected: an API server and a command line tool for accessing the API. The API should follow standard REST design with json input/output.

GENERAL REQUIREMENTS

1. The project can be delivered in Java, C# or Go. Other languages are allowed with prior approval.
2. Third-party frameworks and libraries are allowed but their usage must be documented.
3. Both the API and command line tool should run on Windows or Linux.
4. Please attach detailed instructions and guidelines for building and running the application.
5. All documentation needs to be attached in a separate document file and properly formatted in Markdown syntax.

6. Make your application available in a public GIT repository of your choice: GitHub, BitBucket, GitLab, etc.

COMPONENTS

Component 1: API

This is a REST-based API that will be called by the command line tool. It will wrap the Marvel API to provide custom filtering, sorting, note management and simple analysis.

Required functionality:

- All requests and responses must be in json.
- API to return a list of the creators with the following functionality:
    - The fields to include: id, fullName, modified, the custom note (if any), # of comics, #of stories, # of series and # of events
    - Sorting and filtering on those fields
    - Additional filtering by name or id a comic, series, story or event
- API for the management of custom note for creator - this is a single note on a creator:
    - Notes can be stored in a local file
    - List current notes with filtering by id or text. Response should include id and name of the associated creator
    - Ability to create, edit and delete the note
- API for performing a simple comparison between two creators. Given two creator ids, return the following data:
    - A summary for both creators containing:
        - id, first name, middle name, last name, modified date
        - Any custom note
        - # comics, # series, # stories, # events for the creator
        - # comics, # series, # stories, # events in common with the other creator
    - A list of all comics, series, characters, events shared between the two creators
        - Should include an id, type and name

Component 2: Command Line Tool

The goal of the command line tool is to provide an interface to the API. The exact syntax of the commands are up to you.

Required command:

- Help - returns a list of commands and their syntax

    - Parameters: none
- List - shows the lists of creators
    - Parameters: filter by X and sort by Y
- Add note - adds a note to creator
    - Parameters: id and note
- Edit note - updates the note
    - Parameters: id and note
- Delete note - deletes the note
    - Parameters: id
- Compare - displays the creator comparison
    - Parameters: two ids
- Test - runs tests on each of the commands
    - Parameters: test to run or run all tests
    - If preferred, a dedicated testing framework can be used (see next section).

Bonus Functionality (Optional)

The following to-do items are all **optional**. Feel free to include them if you desire.

- Deploy the API service to a cloud-based server. If this is done, there should be an option on the command line tool to connect to either the remote server or the local one.
- Better data storage for the notes. This can be a local or remote service. Services of interest would be Redis, Memcached, a sql database or a key-value store. Installation instructions must be included for any service used.
- Ability to run tests on the API using a dedicated testing framework.

DEADLINE

We are giving a week from the moment you receive this message to develop and deliver the requested app. If unexpected non-related issues arise that prevent you from delivering on time, please let us know immediately. We will favor quality over quantity, so if you happen not to deliver all the functionality proposed, that’s just fine, although we will appreciate a small explanation of the reasons you were unable to complete the full code test.

QUESTIONS?

We will appreciate your proactiveness on this challenge, and we will carefully evaluate how far you can get with no further input from us. All in all, we do understand that things might not be clear at first or that you may like to discuss different aspects of the implementation.

If crucial impediments arise and you find yourself blocked at some point, feel free to contact our Back-End Lead Architect at jonah@weareblueorange.com.