Bridgit Developer Challenge: “screechr”

Scenario

Congratulations! You’ve just been hired as the first back-end developer at what’s sure to be ‘the next big thing’ on the internet: *screechr*.
screechr is going to be a multi-platform social media app that just happens to be a lot like Twitter; except you screech, not tweet. Revolutionary stuff.

Your most important task is to implement the REST API system that will allow the client developers to setup a test environment and test & demo their clients running on iOS, Android, and web. The team has agreed on the following constraints:

- For the purposes of this demo data does NOT need to be persisted between runs and in-memory data storage is fine
- The API is to be a RESTful web service
- The solution produced should run equally well on a Mac or Linux OS *(bonus points for supplying a buildable and runnable docker container!)*
- A script may be provided for running the solution, or the README.md file should contain step-by-step instructions
- If multiple containers are used in the solution, Docker Compose is to be used

Prior to your arrival, the CTO of *screechr* decided that any (or all) of the following languages & frameworks would be OK to use:

- Latest .net Core/ASP.net Core
- Latest Go

The CTO stated she doesn’t care if the demo is a monolith or a collection of microservices so long as it can be easily deployed and run.

Authentication

The CTO has informed you she hasn’t decided about how authorization or authentication will work just yet but that you’re to go ahead with using a simple token-based authentication mechanism that validates a “secret” token value for a given user passed in via the Authorization header.

Authorization

At this point any user can call any endpoint. However, the following rules should be applied:

- A user may modify only their own profile
- A user may modify only their own screeches
- All users (including unauthenticated users) may view all screeches
- Authenticated users may view all profiles

Object Models

For the purposes of the demo-ready API the *screechr* team has agreed that the following object models will cover their needs when implemented at the database level. They’re leaving it up to you to decide what and how these values are exposed at the API layer (including naming).

User Profile

The profile of the user. Contains the following attributes:

- Id – positive integer, upper limit >10 billion, uniquely identifies the user in the system, cannot be null.
- User name – The public user name of the user. 80 characters, unique, cannot be null or all whitespace.
- First Name – The first name of the user. 100 characters, cannot be null or whitespace.
- Last Name – The last name of the user. 100 characters, cannot be null or whitespace.
- Secret token – String used to validate requests as coming from a specific user for this demo. 32 characters (single byte OK), unique and cannot be null.
- Profile Image - Just a URI to a profile image. Can be null.
- Date created – The date & time the user was added to the system. Cannot be null.
- Date modified – The most recent date & time the user was added to the system.

Screech

Our version of a tweet – it allows 1024 characters!
A user profile may be related to none, one, or many screeches. Contains the following attributes:

- Id – positive integer, upper limit >1 quintillion, uniquely identifies the screech in the system, cannot be null.
- Content – The content of the screech. 1024 characters, may be all whitespace but cannot be null.
- Creator Id – The user id who created the screech. Not nullable.
- Date created – The date & time the user was added to the system. Cannot be null.
- Date modified – The most recent date & time the user was added to the system.

Demo API Functionality

screechr is just starting up and because of this your CTO and product manager have agreed on the following must-haves for a demo-ready API:

- API returns correct HTTP error codes for successes and failures, but no additional client messaging is required
- The API accepts and responds with only JSON
- Possible to retrieve a profile by its key
- Possible to update a profile picture by itself
- Possible to update an entire profile,
- Possible to return a paged list of screeches
    - Default sort order is creation date in descending order
    - Can be requested in ascending order by creation date
    - Can be filtered to return only screeches created by a specific user
    - Default page size is 50, maximum is 500
- Possible to return an individual screech by its key
- Possible to create a new screech
- Possible to update the text of a screech
- Authorization & authentication rules are applied correctly
- All dates & times will use ISO 8601 format in the UTC time zone
- All character data should be handled as double-byte
- Unit tests are important - the CTO would like to see a few implemented

Code Challenge Instructions

This challenge is meant to be done on your own within the time period you requested. To submit your work, we ask that you push to a public Github repository. The repo should include everything required to run the solution along with instructions on how to do so in the README.md file.

You’re welcome to use any external libraries, tools, etc. you wish. Please simply include references to these dependencies in a “dependencies” section in the README.md file.

Your Time to Shine

We don’t expect you to build out a production-ready version of *screechr* and we don’t want anyone spending more than ~4 hours this challenge. **You are NOT expected to complete every point on the API Functionality list!**. Just do your best and submit what you have.
Maybe you want to showcase your commitment to TDD? Maybe you’ll demonstrate your ability to write SOLID maintainable, extensible code? Maybe you want to wow us with the fact you nailed every requirement in the list? Whatever it is, we’d be happy to hear your feedback about this challenge and what you feel makes your submission stand out. **Please include your comments and/or feedback within your submission email or in a section of the README.md file with the submission.**

Thanks, and good luck!