RED HAT® MOBILE

Overview

This document is divided into two parts:

- Open Ended Questions
- Practical Exercises

Instructions for each part are provided below.

Part 1 - Open Ended Questions

Please answer **at least 2 questions from each of the 3 sections below**:

Section 1 - Design & Architecture

Q1
Could you outline what you would consider to be the important considerations If you were asked to develop an api that could potentially be called by thousands of users across many mobile applications?

Q2
Given a large legacy system, comprised of a set of distributed systems, how would you approach maintaining backwards compatibility while refactoring or enhancing existing apis in order to avoid disrupting live production systems.

Q3
In your own words, explain what you understand microservices to be. When would you consider using them? What are the pros and cons of using them?

Q4
How would you secure communications between Apps on mobile devices and a RESTful API?

Section 2 - Development & Deployment

Q1
If you were working with an application made up of several services that called out to one another in order to fulfill various tasks and apis, how would you approach testing these services, both individually - particularly at the touch points where they called out to other services - and as part of a whole?

Q2
From the perspective of a technical delivery team, can you think of some deployment, management & support considerations for a hosted SaaS product vs. an On Premise product?

Q3
If you were working on a new feature along with several other distributed team members and you became blocked and unable to continue with your work in a productive manner, how would you go about resolving the problem so that you could continue on?

Q4
What do you value in a code base?

Section 3 - General

Q1
Tell us about a Red Hat product or open source project you really like, and why.

Q2
What do you find most exciting or interesting about working in the software development field?

Q3
What do you envisage as the major technological advancements in mobile & cloud computing in the next 5 years?

Q4
In your career to date, what are you most proud of and why?

Part 2 - Practical Exercises

Please complete **either** Option 1 or Option 2 below.

Please provide a link to a GitHub Repo containing your solution(s).

Option 1:
We've got some User records in a JSON file:
https://gist.github.com/jasonmadigan/009c15b5dc4b4eccd32b

We'd like you to design and implement a web-application that has a RESTful API to interact with this dataset, including filtering, persistence and searching. Your program should be well-tested.

Option 2

Question 1
There's a bug in the following snippet. Can you fix it and write a test to ensure it does not regress?

```
// doThing() defined elsewhere

function foo(callback) {
  doThing(function(err, res) {
    if (err) callback(err);
    callback(null, res);
  });
}

foo(function(err, res){
  console.log('Done!', res).
});
```

Question 2

Debug and fix the following code, and provide a unit test to verify it:

```
function remoteMathService(cb) {
  var one;
  var two;
  callOneService(function(err, num) {
    one = num;
  });
  callTwoService(function(err, num) {
    two = num;
  });

  return cb(undefined, one + two);
}

function callOneService(cb) {
  setTimeout(function() {
    return cb(undefined, 1);
  }, 1000);
}

function callTwoService(cb) {
  setTimeout(function() {
    return cb(undefined, 2);
  }, 1500);
}

remoteMathService(function(err, answer) {
  if (err) console.log("error ", err);
  if (answer !== 3) {
    console.log("wrong answer", answer);
  } else {
    console.log("correct");
  }
});
```

Question 3

Can you modify the attached code to add an exponential backoff so the db is connected to when it is ready i.e. keep trying until it successfully connects

```
// This is a mock database implementation with just a connect function
// db.connect will need to be called a total of 10 times before it successfully
connects
var counter = 0;
var db = {
  connect: function(cb) {
    console.log('connection attempt', counter + 1);
    if (counter < 9) {
      counter++;
      return cb('db not ready yet');
    }
    return cb();
  }
};

// Try to connect, log a successful connection & exit
// If we fail to connect, log an error and return
db.connect(function(err) {
  if (err) {
    console.error(err);
    return;
  }

  console.log('successfully connected!');
});
```