JUAN A. SERRANO

PRESENT ADDRESS
San Francisco Avenue 10th
Badajoz 06002 Spain
(+34) 636110176 - cannyedge34@gmail.com

THOUGTHS LIKE DEVELOPER

This document is something like a "FAQs" that i'm always receiving from all recruiters and companies. So, read it, and if you are agree with me, get in touch!. If you have receiving this document, i'll send you a zip file with code examples (mostly tests). You'll have to take a look to my tests code with these examples...

Question1: Why is this applicant leaving their current role and what are they looking for?

Nowadays, I'm looking for new challenges, English language environment / start-ups with multicultural teams that allow me to grow like developer and learn from the bests ones.

Question2: Outline current salary and details of package and commute

Current Basic: Private Info. Benefits: n/a Notice Period: 15/20 days Current commute time: 30 minutes Flexibility to commute to client site: N/A

Question3: Outline target salary and commute and the thinking behind this

Current Basic: 75.000 Euros Gross. Benefits: n/a Notice Period: 15/20 days Current commute time: 30 minutes Flexibility to commute to client site: N/A

Question4: What frameworks does your current employer promote? What version?

Vuejs/Vuex 2.5 instead of React/Redux. Gitlab Ci instead of Jenkins. Dokku Paas, Docker and Kubernets.

Question5: What was the one hardest thing you had to learn??

1. Tests always must be before of code. If it is not possible because the development time slows down, you have to use a little trick. This trick is... "fuck the test". The test should see it (always always always) Red. If tests are always Green, it's useless.

2. A customer wanted dynamic fields. Yes, it's what you are thinking :S. This customer wanted to create fields in database sent from front-end. It was really difficult convince him to was not a big idea. it was opted for a really complicated fix to make him happy.

3. i had to reinstall two payment systems of 2 customers that used "adaptive payments" from Paypal and processed 1 million euros per month. Paypal left without service and without prior notice to these customers. A lot of money was lost until it was able to migrate to the new Paypal Api.

4. i had to change all tech stack to start using Microservices with json responses with new front-end frameworks instead of Monolithics applications. Create front-end components, create lib/gems to adapt old rails app to new Microservices apps. It was a fucking crazy.

5. i had to migrate one database with 1.5 million of records from postgres to mongodb, being in production.

6. i had to migrate my personal webpage http://2guitarras.com from Linode to DigitalOcean.

7. Advice: Review and refactor the code before sending it to the team leader. Because team leader see things that you'll never see.

8. Advice: Tests must always fail to be sure that the test is correct. Your coverage shoud be greather than 85/90 percent to be sure you are not gonna be big problems with your code.

I've learned a lot of lessons in all these years (TLDR).

Question6: What was the single biggest benefit this framework has brought to your coding?

I was working with react/redux in previous projects and my employer decided to migrate to VueJs 2.5 and was the best decision that was taken. When you need to scale a team easily, VueJs is extremly easy to learn for the new developers and learning curve is 2 or 3 months. React/Redux is a more complex framework and its learning curve is 6 months or 1 year to be learning everything. Also, the code and its complexity is huge compared with Vuejs.

Another important decisions was migrate from Jenkins to Gitlab Ci, use Dokku PaaS to deploy some applications and using Docker and Kubernetes to orchestration.

Question7: What is your current companies approach to developers testing their work.

Testing technologies. I'm using Rspec for Models, Controllers, Services, Integrations, Policies, Jobs, ActionCable and Capybara for front-end. I'm making some tests of front-end with Cypress but i'm having some problems because is a novice framework.

Question8: What testing did you perform? TDD?

This is the TDD best practice that i'm using in my daily life. Summary:

1. Write a small test. 2. Run the test and fail it. 3. Write the Project Code. 4. Run the test and pass it. 5. Make the code pretty (Refactor).

Each step is simple but necessary and essential!

Vital steps to make the test:

1. Arrange. Prepare the environment with variables from factories, before, double, let... 2. Act. Recreate the action of the real code in the test. 3. Assert. Check the test results with expect().to

- There are sometimes the test will pass well if is executed in isolation, but if you run all the tests in the file, it will fail. You should be sure that tests are passing in both ways. Sometimes you'll need clean old objects from tests with after do end to make tests work.

- Other times you will see that the tests pass on your computer however they do not pass in the CI. It happens with the front-end and Capybara. This is because of the servers where CI is hosted, is slower than your computer.

- Use the "driver: :chrome" tool to view the behavior of application with Capybara if you are having problems with front-end tests. Also, it's useful to make snapshot in CI because sometimes one test is failing but you don't know, what was wrong. CI Capybara Snapshots is a great tool to find out what's going on with your front-end.

Random data should never be used in the tests because of tests will fail randomly. Factories must be perfectly defined and clear. Seeds.rb file must be faultless.

Question9: What other type of testing could you employ and why would this benefit?

I'm feeling very comfortable working with rspec for Unit tests (models) and controllers and Capybara in feature/integration tests. To work requests tests, (calling 3rd party APIs) i'm using rspec with libraries like httpie/webmock. I'm be able to work with jest, enzyme, and cypress too, instead of Capybara. I don't need Cucumber with these tools.

Question10: When starting a new feature, what type of test do you start with?

Get in touch with me! and i'll show you feature/capybara tests examples.

Question11: In what order they would write when building a product listing page (like amazon)?

1. Model Product tests (Unit tests) product_spec.rb. 2. Controller tests products_controllers_spec.rb.
3. Features features/products/index_spec.rb.

Get in touch with me! and i'll show you all kind tests examples.

Question 12: How often do you test first versus test after versus don't test?

I'm always trying to write the test before writing application code. But there is sometimes that is inevitable to write the test later because the client wants to finish the sprint ahead of time and the deadlines are very tight. Customer doesn't understand why tests are so important.

Question 13: What approach to test doubles do they take, Fake, Stub or Mock?

I'm using Stubs, Mocks and Spies. Usually, I'm not using Dummies and Fakes.

In my opinion, Stubbing, Mocking and Spying are techniques rather than tools. Specifically, theyre different ways of making assertions while using test doubles.

- Double is an auxiliary RSpec method that creates an object with certain properties that i specify.

- Stubbing: I want to verify a property of the return value of a method, i use stubbing to provide a consistent value to our test subject, so i'm be able to do make a clear assertion about the value it returns.

- Mocking: I want to verify a property of the communication between two objects, i'm using mocking to make assertions about the messages our test double receives.

- Spying: when i want to verify the same communication property as before, but make our assertions at the end of the test, i'm using spying.

Final question. What should I use: a stub, a mock, or a spy? is proving a hard one to answer, perhaps ask: what do I want to verify in this test? instead. Get in touch with me! and i'll show you all kind tests examples with Stubs/Mocks/Fake...etc.

If you received my tests examples files, you can take a look how am i using stubs, mocks and spyes in theses examples.

Question 14: If they use frontend frameworks like Angular/React/Vue.js

- What are their testing strategies here? What types of tests do they use? What don't they test?

At this moment I've been trying to make tests of Vuejs apps, with Cypress Framework, but it's not mature yet. I'm having some problems. I've been testing React app with Jest and enzyme, but in my last project i've been using capybara (too slow) but very safe. All of them are specific frameworks for frontend to test components because of to be testing React/Vuejs components can be a difficult task. Jest allows to write unit tests using the Snapshot feature. Enzyme allows to write unit tests using regular assertions. I've configured gitlab-ci file to take spanshot if capybara features tests are failed. In this way, i'm pretty sure to know what was wrong. All my productions projects are tested with capybara until today, it's more than enough for me. I've never used angular.

Question 15: Scrumb methodology!

- Questions:

1. Knowledge / experience of Scrum and Kanban ceremonies.

2. Where did you last work in a scrum environment?

3. What ceremonies did you part take in and what input did you have at each stage?

4. What are the advantages of Scrum and what are the disadvantages?

I'm going to make a summary of all these questions, describing my day to day.

1. Before estimate sprint duration, all team developers read all stories that product designer give us. It's required read very well all stories because otherwise you are going to get in troubles when are making the estimation.

2. We are making estimation 1 day or 2 days, it depends on how long sprint is.

3. We have a deck of cards in the meeting room. These cards are marked with the fibonacci serie. 1, 2, 3, 5, 8 and so on.

4 The entire team makes estimation (Developers, Designers, QA and Product Designer) all the stories and each developer makes an estimation until an agreement is reached.

5. The team leader distributes the sprint stories to each developer once estimation is finished.

6 Each sprint lasts between 3 and 4 weeks. When sprint starts, all the team communicates through (slack/mattermost) to discuss the doubts that arise during the sprint.

7. Once the sprint has finished, a retrospective is done. It is a meeting with all the team members involved in the project. In this meeting, we discuss what has been done well and what has been done wrong. In this way, we learn from the mistakes we have made.

This is my day to day with scrumbox methodology :).

The disadvantages that i can see with this work methodology, is that you have to meet deadlines and sometimes from product department and the client have not stories well defined and you (like developer) have to deal with the problems and doubts that arise when you are developing the stories. If the stories are not well defined, deadlines will be delayed.

What is your current DevOps Experience? Including CI/CD

In my current company, systems department is responsible for everything related with deployments (Docker y Kubernets). For my personal projects, Capistrano (a long time ago), Dokku and Gitlab-CI. Dokku to do the deployments in production. Gitlab-CI and Dokku are excelents tools to work with and make great deployments in dev branch, pre-production and production (master branch). If you need deploy a big project with a lot of budget, you will need use Docker/Kubernets for orchestration.

What is your experience of Pairing and how do you approach pairing?

Since we use (slack/mattermost), i don't sit down next to a partner to be watching what code he writes, when I have a question/doubt all coworkers are connected to one channel and we try to help each other. In this way we do not waste time. i was some years doing pair programming when these applications (mattermost/slack) didn't exist.