NEKST® IT
TESTING EXCELLENCE COMPANY

WHITEPAPER

Salesforce Functional Testing

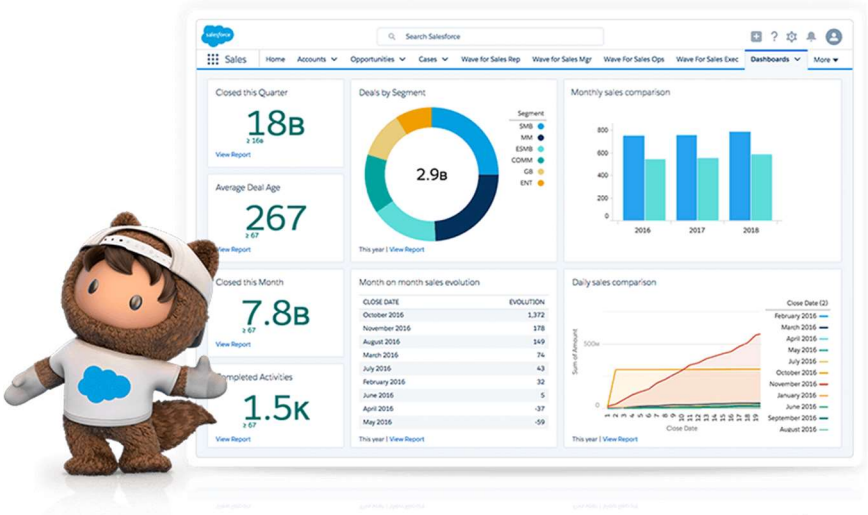
INTRODUCTION

It was early 2014 and I got hired as an agile tester in a project that was implementing Salesforce on a global scale. Back then I didn't know anything about Salesforce, but I knew a lot about testing (close to 10 years' experience, nowadays 16).

I started with some Google queries on best practices, tips 'n tricks and other helpful info on how to test Salesforce, but I found surprisingly little. That's little when it comes to functional testing, since the Apex Developer Guide is quite extensive when it comes to this subject: from best practices on how to get to the forced 75% code coverage to a Stub API, developers are very well taken care of. But all of that is unit tests and only covers code related testing, no functional testing of the configured Salesforce parts.

Now, 6 years of functional Salesforce testing, a Salesforce Admin certification, and multiple projects later, I'd like to share some of my experience and hopefully prevent you from making the same mistakes & assumptions I did.

Dimitri Fioole
Chief Operating Officer Salesforce



NEKST IT BV

Westluidensestraat 55
4001 NE Tiel, The Netherlands

+31 (0)85 303 5697
salesforce@nekst-it.com

CHALLENGE EVERYTHING!

One of the key features of Salesforce is the out-of-the-box and custom code combination. Or maybe it's more out-of-the-box versus custom code. Both have their pros and cons, which depend on a lot of variables which can even change over time, with new and updated features introduced by Salesforce. Not even considering new requirements coming from the business.

Basically, a Salesforce configurator configures the functionality, objects and other features that are in the Salesforce "Box". A Salesforce developer can build custom features and functions that are not part of Salesforce (yet) or modify standard feature just to give it that extra zing.

As a result, most of the time the features delivered by the configurator or developer are working fine by itself, but it can be a different story once you start testing it from a functional perspective, let alone a UAT. A configurator might not have considered the impact on the custom code part and vice versa, simply because both sides lack that knowledge. Here the tester comes in, from the moment you start discussing a feature in your (agile) team, challenge the solutions brought in by the different team members.

The reason to do this is that people generally approach problems and challenges with the tools they master (in this case configuring or coding) but as a result, do not always consider the bigger picture. As a tester you're usually working closely together with the business analyst and end users, which gives you the advantage of assessing the solution as a whole and putting single features in a perspective that adds most value to your solution. Often, I found that the combined ideas that were sparked by the input from every team member's first ideas, were most valuable and usable!

Another advantage of continuous challenging is less bugs in a later stage of development. To me there's nothing more annoying than testing a combination of functions that were all developed separately and seeing it fail at the first (integration/end-to-end) test case. Also, for a configurator or developer this will be a big plus, since who wants to rebuild the same thing repeatedly?

NEKST IT BV

Westluidensestraat 55
4001 NE Tiel, The Netherlands

+31 (0)85 303 5697
salesforce@nekst-it.com

INDIVIDUAL TEST USERS

Sometimes, during the testing, you'll encounter issues that aren't reproducible. Clicking on a button or link on the screen might open a completely different page than expected or give a nasty error without any clue what could've caused that. Given the non-reproducible nature and the fact that it isn't occurring all the time, investigating this is a challenge. It could very well be something with your test user accounts.

In one of my projects, we created test users in all our environments at the beginning, and since I was the only tester, I would be the only one who would use those accounts. After a while, we decided that our configurators and developers should unit test their new features before releasing it to test, to improve the quality throughout our whole development process. What we didn't consider during that decision is which users they'd use for those intakes. Of course, they ended up using the test user accounts, which is no problem for as long as you are not logged in at the same time, but it is a problem if that user account is used by multiple logins at the same time. Salesforce gets confused and as a result all your test results are corrupted.

From that point onwards, everyone had to create his or her own individual test user. This will save you a lot of time investigating false positives!

BROWSER BANTER

One of the things that I've noticed a lot over the years is that not all companies support Chrome. Usually, a Microsoft based browser edge or IE is the standard and can be complicated to deviate from as a company. Therefore, it's important for you as a tester to make sure you're executing all your testcases in the browsers that will be used by your end users in production.

Also make sure you enable the development tools/extensions/options in browsers. Firebug, Chrome apps & extensions, developer tools, etc. They can really help you in the debugging process.

One of the browser features I use the most: private browsing. Sometimes you want to make changes to your test data or Salesforce configuration while testing. Usually this can only be done with an Administrator account, while your test execution is done with an end user (test) account. To prevent you from the hassle of logging off, change settings and logging in again, you can use private browsing. Just open two separate browser windows (not tabs!): in one you log in with the Administrator and in the other one with your test user. This will make your life as a tester easy!

NEKST IT BV

Westluidensestraat 55
4001 NE Tiel, The Netherlands

+31 (0)85 303 5697
salesforce@nekst-it.com

PROCESS AUTOMATION

One of the things where testing really ads value is in the process automation. No matter if it's the Process Builder, Flows, Visual Force or Apex, all of these combine different pieces of functionality and features which automatically implies more risk. First make sure to test all the small bits and pieces, to make sure that's all fine. Once that's done, design your tests based on the business processes and test the process flows from an end-to-end perspective. Not only can this end-to-end perspective be based on the business processes, but also on, for example, the process builder flow.

I hope this brings you some insight for when you are working on a Salesforce implementation. Of course, there's more things I've learned over the years, and I will dig into that in two other blogs: Integration testing & Test Automation.

ABOUT NEKST IT

Nekst IT is the first Salesforce Partner worldwide that focuses solely on Testing & Quality Assurance. This entails a different focus than developing, configuring, or implementing. We fully stand for impartial monitoring of the customer systems' quality.

Nekst IT people are positive, proactive professionals. People who dare to ask questions, investigate, and are not slowed down by the established order. With this attitude, we perform excellently for our customers. Nekst IT test professionals are above average versatile in the test profession. Based on this broad expertise, their test talent is used in challenging projects for various clients.

Do you want to know more about Salesforce testing? Then don't hesitate to contact us. Send an email to **salesforce@nekst-it.com** or give us a call on **+31 (0)85 303 5697**.

NEKST IT BV

Westluidensestraat 55
4001 NE Tiel, The Netherlands

+31 (0)85 303 5697
salesforce@nekst-it.com