Benefits of automated testing

Benefits of automated testing An Abstract Technology experience report on Cypress and BackstopJS related to one of our Plone project Onkopedia.

A working code is the most important element in every web project, be it large or small. Especially in large-scale projects it becomes a challenge to keep the overview. In order not to have to test everything each time a change is made, it makes sense to automate this step. It is a time-saving alternative because it is much faster than manual testing and can be performed at any time.
Automated testing is accurate, which on the one hand ensures the quality of the code and on the other hand creates reliability. At the same time, it also means that the code is protected against accidental cracking, as error notifications are sent when something is not correct.

We implement automated tests in all major projects, using different tools.

_ WHY WE CHOOSE CYPRESS FOR END-TO-END TESTING?

For end-to-end testing we choose Cypress, an open source all-in-one testing framework based on JavaScript. We thought about using Cypress while developing and implementing a faceted search for the client. Another significant and decisive benefit of Cypress in this regard is, that it is entirely built on JavaScript and executed inside the browser itself where it uses the same run-loop as the referring web app. Via Node.js it reads and emulates web traffic and interactions with the browser.

Cypress offers the possibility to write different types of tests, among others end-to-end tests and integration tests. The fact that it is based on JavaScript makes it easy to understand, create and perform tests for both, experienced quality manager and frontend developers. Also debugging is very easy since it provides a precise and straightforward error analysis of single objects. Additionally, screenshots can be taken after each test run, making it possible to reproduce a specific command or state.
All these benefits were decisive for us to use Cypress as an automated testing tool in the Onkopedia project.

How to install Cypress?
All you need to do to install Cypress is to execute npm install cypress. This will set up all dependencies, libraries, engines, servers, frameworks, etc. Further, no configurations need to be made, which does not exclude the possibility to add other libraries, dependencies or settings.

Writing tests in Cypress
As a first step the default setup cypress.json needs to be modified, for example by changing the baseUrl to the referring application address (e.g. localhost:3000). Running cy.visit(url) Cypress prefixes the URL sent with the baseUrl.

After all modifications are done the first file in the integration folder has to be created to actually start testing the environment

```
it(“test if the url has 2 params document types”, () => {
  cy.visit(
    “#https://app-test.onkopedia.com&document_type=drug-assessment||drug-factsheet”
  );
  cy.get(".cards >div").should("have.class", "Mui-expanded");
  cy.get(".cards >div").should(($div) => {
    expect($div).to.contain("Afatinib (Giotrif®)");
  });
});
```

Running the test is quite easy since it requires only to change to the run command. After doing this all the test results will be shown in the terminal. Videos and screenshots will be automatically saved. Those are useful for debugging.

By running the command npx cypress run cypress opens a modal and displays the defined file. When double-click on the file which is to be tested, a Browser tab opens showing the results of the performed test.

_ WHY WE CHOOSE BACKSTOPJS FOR UI TESTING?

To test the UI of our web projects we choose BackstopJS, an open source CSS regression testing framework. It is a simple-to-use tool which automates visual testing by comparing DOM screenshots of pages through various viewports. The changes are highlighted so that the tester can quickly identify and assess them.

The project Onkopedia has grown very quickly over the last few years, and new functions have been developed and implemented. The more functions, the more difficult it became to keep track of all dependencies inside the project. It has happened repeatedly that something has broken without really noticing it.
To avoid losing time fixing bugs that were not discovered or were discovered on the staging server but blocked further production versions until they were fixed, we responded quickly. With BackstopJS we found a tool that automates and facilitates user interface testing.

How to install BackstopJS?
Once installed a project can be set up by running backstop init. This creates a project folder in the referencing working directory and a default configuration file(backstop.json)in which the project-related information can be entered. Once the set up is ready the tests can be written.

Writing tests in BackstopJS
Writing tests in BackstopJS requires first the definition of the different viewports of the devices you want to test (among others the most common are Desktop, Tablet, smartphone, etc.) as shown in the code block below:

```
"viewports": [
      {
         "name": "phone",
         "width": 320,
         "height": 480
      },
      {
         "name": "tablet",
         "width": 1024,
         "height": 768
      },
      {
         "name": "tablet-vertical",
         "width": 768,
         "height": 1024
      },
      {
         "name": "Desktop",
         "width": 1860,
         "height": 1024
      }
   ],
```

As next step, the scenarios have to be specified. This is done as followed: Add the URL you want to test. This can be a local staging instance. Specify the reference URL. This is usually the productive environment of the project.
Define the time the tool should wait before starting the tests (delay). The specifications will look like this:

```
"scenarios": [
      {
        "label": "Onkopedia home page",
        "url": "https://dev.onkopedia.com/de",
        "referenceUrl": "https://www.onkopedia.com/de",
        "delay": 1000
      },
      {
        "label": "Onkopedia guidelines overview",
        "url": "https://dev.onkopedia.com/de/onkopedia/guidelines",
    "referenceUrl": "https://www.onkopedia.com/de/onkopedia/guidelines",
        "delay": 1000
      },
      {
        "label": "Onkopedia drug-assessment overview page",
    "url": "https://dev.onkopedia.com/de/drug-assessment/guidelines",
    "referenceUrl": "https://www.onkopedia.com/de/drug-assessment/guideliness",
        "delay": 1000
      }
 ],
```

In preparation for the tests, the reference screenshots must first be created. The command backstop reference will generate screenshots for every reference URL (productive environment) defined in the scenario specifications.
After having the reference screenshots the test screenshots which will be compared with the created references can be generated with executing the >code>backstop test. When the software finishes the generation of the screenshots the browser window will open and display the referring report. This may look like the following:

Running backstop approve will replace the reference with the new screenshot. This command should only be run whenever an error is spotted but the local/staging version is correct.

_ ABOUT THE PROJECT ONKOPEDIA

The DGHO contributes to the education and training of doctors and medical professionals by developing, producing and publishing medical guidelines. The focus is on research, diagnosis and treatment of blood diseases and malignant tumors. Onkopedia is a project of the DGHO and other medical associations from Germany, Austria and Switzerland and acts as a portal to make the guidelines available to medical professionals as well as patients.
In 2010 the Plone based platform was developed by the company ZOPYX. Abstract Technology was consulted as a project partner to optimize the portal regarding frontend design and usability.

_ OUR CONCLUSION

Automated testing became crucial for us, especially in large projects. Not only to facilitate processes, but also to continue to guarantee the quality of our work. We are very satisfied with the Cypress and BackstopJS frameworks. Together they cover all our requirements especially in the Onkopedia project. They are easy to install, to customize and to handle for both, non-experienced