# Automation Tools for Web Testing. Beyond Selenium

Rosario Segovia Gonzalez[1], Dana K. Urribarri[1,2,3][0000−0001−5446−7077], and Martín L. Larrea[1,2,3][0000−0003−3067−464X]

[1] Department of Computer Science and Engineering, Universidad Nacional del Sur (UNS), Bahía Blanca, Argentina
[2] Computer Graphics and Visualization R&D Laboratory, Universidad Nacional del Sur (UNS) - CIC Prov. Buenos Aires, Bahía Blanca, Argentina
[3] Institute for Computer Science and Engineering, Universidad Nacional del Sur (UNS) - CONICET, Bahía Blanca, Argentina.
rosegovia22@gmail.com, {dku, mll}@cs.uns.edu.ar

**Abstract.** In today's fast-paced world of software development, automation test tools proved to be of great importance in building a robust product while delivering quality software faster. To keep up with ever-changing demand, organizations are embracing test automation. The goal of automation test tools has shifted from shortening test times to better coverage and effective use of test cases. Selenium is a familiar, long-standing, and well-known name in the market of test automation. It is considered the industry standard for web application user interface automation testing. But as it happens most of the time, it also has certain limitations. There are other test-automation systems with good features available in the market. The goal of this article is to review and analyze them as a Selenium alternative.

**Keywords:** Test Automation · Selenium · Cypress · Robot Framework · Cucumber.

## 1 Introduction

Software quality has become a critical determinant of a product's or company's success. As stated by Oztemel et al., [19] "...the superior quality of the manufacturing industry strictly depends on its high quality applied production technology..." and "...there are now companies having the largest part of businesses in their sector with only running a software...". Lee et al. [15] also highlighted the role of software in the industry when they stated that "The Fourth Industrial Revolution is ubiquitous and will increasingly transform and reshape operations/production, supply-chain, management, and governance as well as products and services. Whatever could be codified in the organizational life will be put into codes and software and embedded into cybernetics systems that will replace human work activities". The success of a software product rests on its quality.

Automation test tools are critical for developing a solid product and delivering high-quality software quickly. Selenium [28] is a familiar, long-standing, and well-known name in the market of test automation. It is considered the industry standard for automation testing on the web, but it has certain limitations. There are several alternatives with good features available in the market. These tools are the focus of this article. Our goal is to review and evaluate them as feasible solutions for testing automation.

The rest of the article is organized as follows; we start with a brief presentation of the benchmark used for this work, as well as Selenium in Sections 2 and 3. Then, each software is used to perform the benchmark test cases in Sections 4 to 6. The results of this experience are analyzed and discussed in Section 7 and 8. This work summarizes the one carried out by the main author in her thesis for the Degree in Computer Science. The full report can be found here [8].

## 2 Technologies, Tests and Environment

Selenium is the industry standard for web automation testing. However, it does have some restrictions. To compare Selenium against other tools, we present a test plan. We execute that test plan using Selenium, and then, we will discuss how the resolution using the others tools compares against Selenium.

We started working with Selenium since it is the tool most frequently used for test automation. Selenium works for both functional and regression tests. It provides a solid baseline to compare against other testing frameworks. The others selected tools were Cucumber [9], Cypress [4], and Robot Framework [13]. For this selection, we took into account that they are among those with the highest usage percentage at present, as detailed in [26,22,24,21,10,6,23].

Cucumber is an open-source tool designed on the concept of BDD (Behavior Driven Development). It performs automated acceptance tests by running tests that best describe the application's behavior. Cypress is a framework that includes assertion libraries, mock libraries, and automatic end-to-end tests without using Selenium. It consists of an architecture that executes the commands in the same execution cycle as the application. Cypress runs a Node process that constantly communicates, synchronizes, and carries out tasks. It accesses both the front-end and back-end of the application and responds to events in real-time. Robot Framework is an automation system that executes the keyword-based methodology for Acceptance Test-Driven Development (ATDD) and user acceptance testing.

### 2.1 The test object

The test object for this research was an internal web application used by a telemedicine and virtual medical care company. This website has the name *Fms Web Portal*. The purpose of the web is to allow employees to manage several assets within the company.

### 2.2 The test plan

The test plan for front-end validation consists of two sets of tests. The first set is for the validation of a field. The second one is for the validation of communication between sections. The first set contains test cases for allowed characters (RF_01), prefixes not allowed (RF_02), unique username (RF_03), and permitted length (RF_04). The second set contains the RF_05 case which verifies that the default value of the checkbox is linked to the value selected in another section of the system. For the verification of the back-end, the test plan includes tests for the verification of the correct storage of the user in the database (RF_06) and that all the fields are stored correctly (RF_07).

### 2.3 Experiments availability

To facilitate the repetition of this work, the complete test plan[4] and all the experiments' source code are available online[5].

## 3 Selenium Web Driver

To establish a solid baseline for the comparison of different test automation tools, we provide an introductory summary of the main features and some details about Selenium. Selenium is a test automation framework used to validate web applications on different browsers and platforms.

It was born as a Firefox module or add-on for testing web applications within the browser. It is currently a tool that is being used by most of the software automation market. Selenium allows, on the one hand, to remotely control different instances of a browser and emulate the interaction of a user with the browser; on the other hand, it allows users to simulate common activities performed by end-users such as entering text into fields, selecting values from dropdowns and checkboxes, and clicking links in documents. It also provides many other controls such as mouse movement, arbitrary JavaScript execution, and so on.

Selenium Web Driver is a collection of APIs used to automate the testing of a web application. Selenium supports the automation of all major browsers on the market. It defines a language-neutral interface to control the behavior of web browsers. Each browser uses a specific implementation of the Web Driver, called a driver. The communication is bidirectional: The Web Driver passes commands to the browser through the driver and receives information through the same route.

The main job of Web Driver is to handle communications with the browser. Web Driver is unaware of testing concepts, such as comparison results, verification of assertions, or validation. At this point, other testing frameworks come into play such as XUnit [32] or NUnit [20] for .NET, JUnit [2] for Java, RSpec

[4] `https://drive.google.com/drive/folders/1V8KVlVB1FzT_yaXUG-7hTQMRBca4QW9k`

[5] `https://github.com/rochy22/UNS_Testing_Automation_Tools`

[1] for Ruby, etc. The testing framework is responsible for executing the Web Driver and related steps of the tests.

This tool is not without limitations. It has a limited testing environment because it only supports the testing of web-based applications. Mobile applications, Captcha, and Barcode readers can not be test using Selenium. The only way to generate reports is by using third-party tools like TestNG [3] or JUnit [2]. The user must have prior knowledge of programming language. Selenium does not have tools for image-based tests; this is supported by other software such as Sikuli [14]. There is no native reporting feature in Selenium, but it is possible to integrate Selenium with TestNG [3] or JUnit [2].

### 3.1 Using Selenium with our Test Plan

To facilitate the reading of the implementation carried out, we generated three classes called: Specialties, Database, and Username, which cover all the cases belonging to the test plan presented. The implementation of the tests that cover RF_05 is found in the Specialties.cs file. The implementation of the tests covering RF_06, RF_07 is found in the Database.cs file. The implementation of the tests covering RF_01, RF_02, RF_03 and RF_04 is found in the Username.cs file.

We used the DbContext class, provided by Entity Framework Core [16], to capture and verify the data storage. The DbContext class is an integral part of the Entity Framework.

## 4 Cypress

Cypress allows the creation of end-to-end tests, integration tests, and unit tests. The framework includes assertions, mocks, and automatic e2e tests libraries without using Selenium. Unlike the latest, it has a new architecture, which executes the commands in the same execution cycle as the web application. This solution is most often compared to Selenium; however, Cypress is fundamentally different. Selenium works by running outside the browser and executing remote commands over the network (Libraries + WebDriver). Due to this structure, Selenium works through certain servers, which can cause delays under some circumstances. The Cypress engine operates directly within the browser, allowing it to listen to and modify browser behavior at runtime by manipulating DOM and altering network requests and responses on the fly.

### 4.1 Strengths

The main advantages of using Cypress are ease of installation and a quick start of automation. Cypress has several built-in tools that allow the user to start the automation immediately after installation. The software records the actions carried out for better debugging. It can take screenshots and videos. Its dashboard allows the tester to create reliable reports. There are well-described documentation and examples on the official site. Functionality to execute commands in

real-time, offering visual feedback at all times. It has a *headless* mode, which is a functionality that allows us to run our tests in the background, hiding the browser.

### 4.2 Limitations

Although Cypress has advantages over other automation tools, it also has limitations. One limitation is that Cypress only allows interaction with a single tab. Given its architecture and how Cypress runs in the browser, there are ways to mitigate this situation. Cypress does not allow browsing to superdomains, so redirecting the test to two different pages is not possible. It is not possible to mouseover with Cypress. Cypress has an extra monetary cost and this rate increases depending on the number of users. The "Page Object Model" pattern is not recommended by the Cypress developers. Knowledge of Javascript is required for script generation.

### 4.3 Cypress vs. Selenium, round one

Below we break down the differences obtained between both tools based on their available information [29,11,5]. Using Selenium involves a lot of preparatory work, which is quite time-consuming. Cypress is a single framework that allows faster installation and implementation start. Cypress supports JavaScript only, while Selenium is an open-source tool that supports a wide spectrum of programming languages such as C#, Python, Ruby, R, Dart, Objective-C, and JavaScript. This versatility makes Selenium a more suitable choice for QA teams whose scope of work is broader and more diversified. Cypress rules out options like Jest or Tape for frontend developers working with this testing tool. Selenium gives QAs the flexibility to select the programming language of their choice such as Java, Ruby, and Python while Cypress only Javascript.

When it comes to offering choice to its users, Cypress not only limits the language, but also the framework that can be used for testing. With Cypress, the tester can only use the Mocha testing framework to write tests. Mocha's role in JavaScript is comparable to that of XUnit in C# or Junit in Java. Cypress is often considered the faster alternative compared to Selenium. Its Built-in Mocking functionality plays an important role in delivering the element of speed. Cypress mainly focuses on the frontend. They work with simulated HTTP XML requests to the server to speed up the testing process. Cypress offers a Built-in Server Mocking functionality. While it is possible to simulate server responses even with Selenium WebDriver, the process is more elaborate. Cypress does not work with IE and Safari. It also lacks multi-tab support and cannot be used to handle two browsers simultaneously. Selenium, on the other hand, has no such limitations. It can be used to test apps on Chrome, IE, Firefox, Safari, Edge, and mobile browsers.

| | | Cypress | Selenium |
|---|---|---|---|
| Headless mode Inactive | Username | 40.53 sec | 90 sec |
| | Specialties | 39.80 sec | 42.1 sec |
| | Database | 62.26 sec | 59.4 sec |
| Headless mode Active | Username | 41 sec | 102 sec |
| | Specialties | 42 sec | 42.1 sec |
| | Database | 62 sec | 59.4 sec |

**Table 1.** Execution times for the test cases in both frameworks, under the same hardware configuration.

### 4.4 Cypress vs. Selenium, round two

In the same way as with the example case implemented in Selenium, the test cases were separated into three classes called: Specialties, Database and Username, which cover all the cases belonging to the test plan presented.

The implementation of the tests that cover RF_01, RF_02, RF_03 and RF_04 are found in the Username.js file. The implementation of the tests that cover RF_05 are found in the file Specialties.js. The implementation of the tests that cover RF_06 and RF_07 are found in the Database.js file. We used cypress-sql-server to access the database. This tool allowed us to access the information within the database through a query. Making a comparison with Selenium we can see that the use of cypress-sql-server is less readable than the use of DbContext. After the implementation of the example cases, there was a strong difficulty in verifying the data in the database using Cypress.

Cypress is arguably faster (Table 1) in terms of execution time in most cases. The only occurrence in which it is slower is when querying the database. This is understandable since it is a tool created to check the correct behavior of the frontend and not the storage information.

### 4.5 Recap

Although both tools are designed to automate tests, they differ in their purpose, target users, and architecture. Another noteworthy aspect is that Cypress is a new tool that is constantly growing and improving, while Selenium has been an established tool in the field of automation testing. In conclusion, we can say that Selenium is a more flexible tool when it comes to working, but it requires more preparation and more time, unlike Cypress, which is ready to be used immediately after installation, giving it a certain advantage in some situations.

## 5 Robot Framework

Robot Framework is a keyword-based automation framework for Acceptance Testing, Acceptance Test-Driven Development (ATDD), Behavior Driven Development (BDD), and Robotic Process Automation (RPA). It applies in distributed and heterogeneous environments, where automation requires the use of

different technologies and interfaces. This tool is actively supported and used by leading companies in the industry in their software development. It is independent of both the operating system and the application. The main framework is implemented with Python and also runs on Jython (JVM) and IronPython (.NET). It is open-source software released under the Apache License 2.0, and most of the ecosystem's libraries and tools are also open source. The framework was initially developed at Nokia Networks and was open-sourced in 2008.

Its infrastructure is made up of four well-differentiated modules or layers [17]. It has a highly modular architecture. Following a bottom-up approach, each of them is briefly described below: System under test, it is the most physical layer of the architecture. Here are any and all physical systems, applications, environments, etc. to be tested, automated. Next is the testing layer, with its own tests and libraries. This layer connects to the previous one through system interfaces and to the next one through an API test library. The Robot Framework infrastructure can be found at the next level. Analyze the test data and interact with the lower layer. Finally, there is the test data.

### 5.1 Strengths

Among the main advantages of using the Robot Framework is [27] that it can be integrated with virtually any other tool to create powerful and flexible automation solutions. It is open-source and free to use, with no license fees. It allows an easy-to-use tabular syntax to write test cases uniformly. The framework provides easy-to-read reports and logs in HTML format. There are options to combine the results of multiple test runs. It is platform and application-independent. Provides a simple API for creating custom test libraries. These libraries are supported natively with Python or Java. Robot Framework provides a command-line interface and XML-based output files for integration into existing build infrastructure (continuous integration systems). Parallel execution is possible with the simultaneous use of the Robot framework and Pabot [18]. It also supplies support for Selenium for web tests, Java GUI tests, running processes, Telnet, SSH, etc. It supports the creation of test cases based on data. It has built-in support for variables, handy for testing in different environments. Provides tagging to categorize and select test cases to run. Allows easy integration with source code control: test suites are just files and directories that can be versioned with production code. The modular architecture supports test creation even for applications with many diverse interfaces. And finally, it has easy integration with Jenkins and Maven.

### 5.2 Limitations

This tool does not have the built-in debugging capability; in other words, the tester have no options for the typical breakpoint arrangement. Amazon web services do not support Robot Framework tests. Alternatively, the SauceLabs portal can be used and all tests will be run on the Robot Framework. It also has IDE difficulties, on some occasions, the tool crashes both in "text editor"

mode and when using tabular mode. Also, it is worth noting the impossibility of working with some third-party plugins due to the lack of autocomplete in some libraries (for example, Selenium Library). The Robot Framework lacks support for nested if-else loops, which are needed when code becomes complex.

### 5.3 Robot Framework vs. Selenium, round one

Robot Framework is a set of programs and libraries to create test cases. The framework creates test suites based on reusable keywords written from other keywords or programming languages. On the other hand, Selenium is a library interface for a driver that controls a browser. A tester can't write tests using only Selenium; the tester needs something like a programming language (Python, Ruby, etc.) or a testing framework (Robot Framework, Cucumber, etc.). Selenium itself does not provide any way to run tests or generate reports.

Using Selenium involves a lot of setup work and takes more time than Robot Framework. On the other hand, tests are easier to understand and implement using the Robot Framework. There are several limitations when working with Robot Framework. All if-else can only be in simple blocks. Nested loops are not possible. These aspects are of paramount importance when performing complex encodings. As a plus for the Robot Framework, the tester can write a custom keyword in Python.

It is very easy to find a tester who knows about Selenium but the percentage of people who have worked with Robot Framework is lower. However, the learning curve is very short compared to Selenium. Also, some programming knowledge is needed, such as basic knowledge of Python if someone wants to write a custom library.

Finally, Robot Framework does not work with IE, Safari, or Opera. Selenium, on the other hand, has no such limitations. It can be used to test apps on Chrome, IE, Firefox, Safari, Edge, and mobile browsers.

### 5.4 Robot Framework vs. Selenium, round two

In the same way that we did with Cucumber, we wrote three classes Specialties, Database and Username. The tests for RF_01, RF_02, RF_03 and RF_04 are available in the username.robot file. The implementation for RF_05 is in the file specialties.robot. The implementation of the tests that cover RF_06 and RF_07 are found in the database.robot file. To access the database we used DatabaseLibrary. This database library contains utilities designed for use by the Robot Framework. This allows the database to be queried after an action has been performed to verify the results.

### 5.5 Recap

Robot framework provides the user with greater ease in writing test cases. This may be observed in the way locators are obtained, functions are written, and the database is accessed. On the other Selenium with Xunit is significantly faster in terms of runtime than Robot Framework in all cases (Table 2).

| | | Robot Framework | Selenium |
|---|---|---|---|
| Headless mode Inactive | Username | 150 sec | 90 sec |
| | Specialties | 75 sec | 42.1 sec |
| | Database | 85 sec | 59.4 sec |
| Headless mode Active | Username | 132 sec | 102 sec |
| | Specialties | 74 sec | 42.1 sec |
| | Database | 66 sec | 59.4 sec |

**Table 2.** Execution times for the test cases in both frameworks, under the same hardware configuration.

## 6 Cucumber

Cucumber is a high-level software testing tool. It is open-source and written in Ruby. The tool allows for writing test cases in a very accessible and easy-to-understand way, regardless of the depth of technical knowledge [7]. Cucumber was developed for the Behavior Driven Development (BDD) model. This model seeks to implement meaningful acceptance test scenarios while development is in progress. To achieve this goal, the approach seeks to write tests that verify the behavior of the code and its correct operation from the point of view of the business team before writing any code.

Some organizations use Cucumber within a Selenium framework. In this way, they enable reliable test automation that emphasizes plain language [30]. Doing so leads to a shared understanding of how the software should work. There is also an improvement in collaboration between testers, coders, and decision-makers. Finally, another benefit of this combination of tools is that web testing can be automated across all browsers at scale.

### 6.1 Strengths

This tool enables continuous collaboration and increased visibility between developers, testers, users, and management. The BDD strategy fits well with agile methodologies since they specify requirements such as user and acceptance stories. The definition approach helps a common acceptance of the functionalities prior to development. The specification and the tests are in the same document and shared vocabulary. This reduces the possibility of confusion when transforming specifications into features or unit tests. Cucumber supports different languages and acts as a bridge between business and technical language. This happens because the user can create test cases in plain English. Additionally, it allows test scripts to be written without programming knowledge, making it easy for non-programmers to get involved. Due to its simple test script architecture Cucumber allows for high code reuse. Finally, it has a quick and easy setup and execution

### 6.2 Limitations

The tool requires tests to be adapted to a predefined format. The additional layers of abstraction used can add time and effort to the team. If the tool is not combined with BDD practices it can lead to frustration. Although it does not require programming knowledge, a basic knowledge of regular expressions is necessary.

### 6.3 Cucumber vs. Selenium, round one

Selenium and Cucumber are similar in some ways. Both are open source and are used for functional testing [25]. However, there are some key differences; in terms of functionality and usage, Selenium and Cucumber are two different things. Selenium is a web browser automation tool. Cucumber is a BDD tool that can be used with Selenium or Appium. Selenium is preferred by technical teams while Cucumber is often preferred by non-technical teams.

In Selenium test scripts can be written in C#, Python, Ruby, R, Dart, Objective-C, and JavaScript while Cucumber has implementations for almost any programming language: JRuby, Java Groovy, C++, JavaScript, Clojure, Gosu, Lua, .NET, PHP, Python, Tcl. In Cucumber the test scripts are written in plain text language according to the rules of the Gherkin language.

The syntax for writing Selenium scripts has similarities to developing an application. Writing Cucumber scripts is like documenting the procedure or functionality in the correct order. For this reason, Selenium scripts are complicated to develop and run, while in Cucumber it is easy to develop and run the test. On the other hand, in Selenium, identifying syntax errors is easy during development, while in Cucumber, syntax errors are not noticeable when typing. Finally, another point to note is that in Selenium you can use conditional statements, while in Cucumber you cannot use conditional statements. Selenium scripting is complex while Cucumber is simpler

### 6.4 Cucumber vs. Selenium, round two

We again wrote the three classes Specialties, Database and Username. The tests for RF_01, RF_02, RF_03 and RF_04 are available in the username.feature file. The implementation for RF_05 is in the file specialties.feature. The implementation of the tests for RF_06 and RF_07 are found in the database.feature file. Cucumber does not provide any tools for database access. For this reason, mssql was used. The purpose of this library is to provide resources for database access for Node.js. This allows the tester to query the database after an action has been taken to verify the results.

### 6.5 Recap

Cucumber provides the test automation developer with a list of user cases ready to automate. However, after that point, the correct and efficient implementation

| | | Cucumber | Selenium |
|---|---|---|---|
| Headless mode Inactive | Username | 1 min 38 sec | 90 sec |
| | Specialties | 49 sec | 42.1 sec |
| | Database | 1 min 09 sec | 59.4 sec |
| Headless mode Active | Username | 1 min 36 sec | 102 sec |
| | Specialties | 45 sec | 42.1 sec |
| | Database | 1 min 36 sec | 59.4 sec |

**Table 3.** Execution times for the test cases in both frameworks, under the same hardware configuration.

of the test cases is linked to the knowledge possessed by the person in charge of said task. On the other hand, an important factor related to execution time is evident (Table 3). Selenium with Xunit is significantly faster in terms of runtime than Cucumber in all cases.

Both selenium and cucumber have strong points to be chosen. For one thing, Selenium is always the first choice for developers to script and test applications. On the other hand, Cucumber is not as famous as Selenium in the market, but it is also slowly and steadily getting its market. It is recognizable that Cucumber has a faster learning curve so that a beginner can easily learn Cucumber as it is easy to understand and write without any difficulty.

Automation is one of the fundamental requirements in software development companies as it helps the team check for vulnerabilities more effectively and consistently. Daily reports are generated to check the progress, which helps the quality of the delivery and more importantly, the delivery on time without any problem. Selenium and Cucumber tools are useful to achieve these main things for any organization in functional areas. Each tool has its own strong areas.

## 7 Discussion

The main objective of the work was to carry out an analysis and comparison of alternative tools to Selenium as a testing automation tool. The study path began by investigating the tools most preferred by a quality control teams. In addition, a web environment was selected and a test plan was defined for the verification and validation of a set of present functionalities with the aim of obtaining a practical approach to each of the selected tools.

We began by researching Selenium, both from theory and practice. The main purpose of this research was to capture its characteristics, with the intention of subsequently being able to detect the reason why it is currently the market leader. The study of Selenium was followed by Cypress, then Robot Framework, to then move on to Cucumber, making with each one its corresponding comparison to Selenium, detecting weaknesses and strengths in different aspects.

This work allowed us to provide arguments both for and against each of the tools studied. It is necessary to recognize that the choice of the tool will depend a lot on the company, the purpose of the project, the knowledge of the person

who is going to use it, the distribution and communication of the team, the area in which it is developed, among other aspects. However, from the analysis and comparison of the results obtained, it is evident that Selenium is not an insurmountable tool, there are other alternative tools on the market that stand out above Selenium in one aspect or another. This leads us to think that an exhaustive analysis is required before choosing an automation tool for a project, which is only possible with a work team with solid prior knowledge and strong foundations in the area of software test automation.

The versatility that Selenium presents through its easy coupling to other tools is its main advantage. With this statement we can understand that Selenium is a conductor and that the tool that will be used at one end will determine the qualities that are sought in a project.

## 8 Conclusions & Future Work

This project arose from the need to answer questions about the preference in the market for the use of a leading software test automation tool. Why is Selenium the leading tool today? What is the differential advantage in relation to other tools? Is Selenium really the best software test automation tool?

This project arose from the need to answer questions about the preference in the market for the use of a leading software test automation tool. Why is Selenium the leading tool today? What is the differential advantage in relation to other tools? Is Selenium really the best software test automation tool?

The perfect software does not exist, hence it is necessary to recognize that the choice of a software test automation tool will depend a lot on the company, the purpose of the project, the knowledge of the person who is going to use it, the distribution and communication of the team, of the field in which it is developed, among other aspects. However, from the analysis and comparison of the results obtained, it is evident that Selenium is not an insurmountable tool, there are other alternative tools on the market that stand out above Selenium in one aspect or another.

This work represents the beginning of a tool analysis path within the software verification and validation area. There are more tools to study; Katalon Studio [12] and Watir [31] two tools used by test automation teams, which would merit analysis and comparison to discover the features that make them stand out. Katalon Studio is a monthly cost tool that opens the research panorama to paid automation tools and the possibility of giving another focus to research by making a comparison between paid tools and free tools.

Another noteworthy aspect is that, as of the date of writing this article, Cypress is a considerably new tool and in a very early stage of development compared to other test automation tools. All the negative aspects and limitations found during the Cypress investigation are a consequence of this. For this reason, it would be interesting to propose a new study of Cypress as an automation tool, delving into its growth possibilities and the effect it could have on the market within a time.

**Acknowledgments** This work was partially supported by the following research projects: PGI 25/N050, and PGI 24/ZN35 from the Secretaría General de Ciencia y Tecnología, Universidad Nacional del Sur, Argentina.

## References

1. Baker, S.: rspect - behaviour driven development for ruby. https://rspec.info/, accessed: 2022-05-05
2. Beck, K., Gamma, E., Saff, D., Vasudevan, K.: junit - testing framework for java and the jvm. https://junit.org/junit5/, accessed: 2022-05-05
3. Beust, C.: Testng - testing framework for the java programming language. https://testng.org/doc/, accessed: 2022-05-05
4. Breiding, C.: Cypress - fast, easy and reliable testing for anything that runs in a browser. https://www.cypress.io/, accessed: 2022-05-16
5. Carlo, A.J., Montaño, R.: Ya utilizo selenium, ¿por qué debería aprender cypress? https://www.encora.com/es/blog/ya-utilizo-selenium-por-que-deberia-aprender-cypress, accessed: 2022-05-23
6. Colantonio, J.: 58 best automation testing tools: The ultimate list guide. https://testguild.com/automation-testing-tools/, accessed: 2022-05-16
7. Cucumber: What is cucumber? https://cucumber.io/docs/guides/overview/, accessed: 2022-05-27
8. Gonzalez, R.S.: Alternativas a selenium como herramienta de automatización para el testing en la web. https://drive.google.com/file/d/1uThbHfQi9ghplS8Sr3IZjeAqxbaaoBm-/view?usp=sharing, accessed: 2022-05-05
9. Hellesoy, A., Wilk, J., Wynne, M., Hnatiuk, G., Sassak, M.: Cucumber - Tools & techniques that elevate teams to greatness. https://cucumber.io/, accessed: 2022-05-16
10. Help, S.T.: Top 20 best automation testing tools in 2022 (comprehensive list). https://www.softwaretestinghelp.com/top-20-automation-testing-tools/, accessed: 2022-05-16
11. John, S.: Cypress vs. Selenium: Which is the Superior Testing Tool? https://www.cuelogic.com/blog/cypress-vs-selenium, accessed: 2022-05-23
12. Katalon: Katalon website - an all-in-one test automation solution. https://katalon.com/, accessed: 2022-04-04
13. Klärck, P., Härkönen, J.: Robot framework. https://robotframework.org/, accessed: 2022-05-16
14. Lau, B.: Sikuli - automate what you see on a computer monitor. http://sikulix.com/, accessed: 2022-05-05
15. Lee, M., Yun, J.J., Pyka, A., Won, D., Kodama, F., Schiuma, G., Park, H., Jeon, J., Park, K., Jung, K., et al.: How to respond to the fourth industrial revolution, or the second information technology revolution? dynamic new combinations between technology, market, and society through open innovation. Journal of Open Innovation: Technology, Market, and Complexity **4**(3), 21 (2018)
16. Microsoft Build: DbContext Class. Sitio web. https://docs.microsoft.com/en-us/dotnet/api/microsoft.entityframeworkcore.dbcontext?view=efcore-2.0, accessed: 2022-05-23
17. Molinero, A.: Robot Framework and test automation. https://www.teldat.com/blog/robot-framework-open-source-test-automation/, accessed: 2022-05-24

18. My Developer Planet: Parallel Testing With Robot Framework. https://mydeveloperplanet.com/2020/09/09/parallel-testing-with-robot-framework/, accessed: 2022-05-26
19. Oztemel, E., Gursev, S.: Literature review of industry 4.0 and related technologies. Journal of Intelligent Manufacturing **31**(1), 127–182 (2020)
20. Poole, C., Prouse, R., Busoli, S., Colvin, N.: Nunit - unit-testing framework for all .net languages. https://nunit.org/, accessed: 2022-05-05
21. Rajkumar: Best selenium alternatives (free and paid) in 2022. https://www.softwaretestingmaterial.com/best-selenium-alternatives/, accessed: 2022-05-16
22. Rungta, K.: 20 best selenium alternatives in 2022. https://www.guru99.com/selenium-alternatives.html, accessed: 2022-05-16
23. Singh, S.: Top 10 automation testing tools in 2022. https://www.netsolutions.com/insights/top-10-automation-testing-tools/, accessed: 2022-05-16
24. Software Testing Help: Top 10 best selenium alternatives you should try. https://www.softwaretestinghelp.com/selenium-alternatives/, accessed: 2022-05-16
25. Sugandhi, A.: Differences between selenium and cucumber. https://www.knowledgehut.com/blog/software-testing/selenium-vs-cucumber-difference, accessed: 2022-05-27
26. testbytes.net: 11 awesome selenium alternatives for testers. https://www.testbytes.net/blog/selenium-alternatives/, accessed: 2022-05-16
27. TestMatick: Robot framework: A quick review, main benefits, and drawbacks. https://testmatick.com/robot-framework-a-quick-review-main-benefits-and-drawbacks/, accessed: 2022-06-09
28. The Selenium Project: Selenium website. https://www.selenium.dev/, accessed: 2022-04-04
29. Unadkat, J.: Cypress vs Selenium: Key Differences. https://www.browserstack.com/guide/cypress-vs-selenium, accessed: 2022-05-23
30. Unadkat, J.: Introduction to cucumber testing framework. https://www.browserstack.com/guide/learn-about-cucumber-testing-tool, accessed: 2022-05-27
31. Watir: Watir website - an open source ruby library for automating tests. http://watir.com/, accessed: 2022-04-04
32. Young, N.: Xunit - unit testing tool for the .net framework. https://xunit.net/, accessed: 2022-05-05