Touchable Code: Building a multi-touch IDE

Carina Neves Fonseca
carina.fonseca@tecnico.ulisboa.pt

Técnico Lisboa, *University of Lisbon*

September 2017

Abstract

OutSystems is a rapid desktop application delivery platform that enables enterprise-grade web and mobile apps. One of the main tools in the platform is an IDE to develop applications, called Service Studio, currently only available as a desktop version. To follow the recent adoption of mobile devices, this article presents an iPad version of the Service Studio, called OutSystems Touch whose goal is to allow OutSystems users to develop and test their applications' user interface continuously on a mobile device, in a user-friendly way. To accomplish this goal, a study of similar software and applications for business purposes is presented as well all steps taken for the development of OutSystems Touch.

1. Introduction

Technology is evolving at an incredible speed and that is changing the way we interact with it. Mobile devices are becoming the prevailing computing platform for most people and consequently, the number of applications available on the market is increasing.

Traditionally, software applications for mobile devices such as tablets or smartphones are developed on the desktop and this poses a challenge for developers. Unlike web apps, mobile apps need to be developed on a device (computer) and tested in a completely different one (tablet or smartphone). This makes the development process longer, harder and more error-prone. Even though there are already some tools that allow the development of applications, games or prototypes using a touch environment without a traditional keyboard, all of them were developed for individual use or for specific use cases and there is no development tool available for mobile devices in the enterprise market.

OutSystems is a rapid application delivery platform that allows the development of enterprise-grade web and mobile apps faster than traditional technologies. However, as other development platforms, it only works for desktop. In order to keep up with this evolution, the main goal of this project is to allow OutSystems users to develop and test their apps continuously on a mobile device, building an iPad version of the OutSystems Service Studio. This will have several benefits such as:

1. By designing the platform for a smaller screen with larger touch targets, the resulting experience will be simpler and easier to learn.

2. As the line between touch and desktop is becoming thinner and thinner, with devices such as iPad Pro and Microsoft Surface, it is very important for companies such as OutSystems to start designing experiences that go beyond the keyboard and mouse.

3. By developing directly in a mobile device, the resulting apps will probably be more suited to the target form factor.

4. The developers will get immediate end user feedback, in the field, and react fast by tweaking the apps in real time.

Even though mobile devices are typically equipped with powerful batteries, graphic processors and high-resolution screens, high productivity applications are still a challenge to develop. Moreover, when developing a mobile version of an existing platform, it's important to keep the same patterns given that people are already familiar with them. OutSystems has many features, and making them available in a touchable environment was a big challenge.

2. Context

Traditional hand-coding methods for software development are too slow to keep up with the rising demands of consumers. As a result, many companies are innovating and providing users with an easy way to develop applications with low-code platforms. Low-code platforms are defined by Forrester as *platforms that enable rapid delivery of business applications with a minimum of hand-coding and minimal upfront investment in setup, training and deployment* [1]. End users can change apps quickly by modifying logic flows and using new User Interface (UI) elements, and developers can also take advantage of new features without the need to invest in infrastructure and software.

OutSystems is the leading low-code platform and it distinguishes from competition for the user experience and advanced features it provides. However, Google recently launched Google App Maker which can change the competitive landscape, because it is an application accessible to everyone (developers or not), so it will be very easy to use. With the close connection to android and material design, and the ease-of-use that is standard in all Google products, Google App Maker will certainly come as a big disruption in the low code market. As a response, OutSystems and the other Platforms will need to innovate in order to maintain their status as leaders in the Low-Code spectrum.

OutSystems provides a complete solution for developing enterprise-grade mobile and web applications. It has a set of components to visually develop apps and automatically generate their code, integrate them with external custom code (JavaScript, Java, C #, SQL, CSS, and HTML), manage the application's configuration and lifecycle and monitor the apps' performance. One of the main tools in the platform is Service Studio, an IDE where users can develop applications. It will be the main focus of this thesis.

Service Studio allows users to develop web and mobile applications with high productivity, integrates with existing systems and delivers beautiful and usable native mobile experiences and responsive web apps. It allows users to create interfaces by providing a set of over 140 designer-approved patterns and controls for an interface with great look and UX by default. More proficient developers will also be able to use custom code in HTML, CSS and JavaScript whenever needed. It is also possible to model custom business logic, as well as to integrate with 36 different external services and applications like Google Drive, Facebook, Paypal, Dropbox, etc., that can incorporates in the visual data workflow.

Finally, it allows users to create a visual data model and later display the data in the UI and change it in the business logic. The generated apps are secure by default, they run fast regardless of the number of users and data volume and can be deployed and monitored in cloud or on-premises environments.

The existent tools that allow users to build applications and prototypes in a touchable environment, they are typically suited for small applications or tweaking larger apps that are mainly developed on a computer. They require knowledge of specific programming languages and very limited visual editing capabilities. Moreover, they are all targeted for the consumer market and there is still no solution for the business market.

3. OutSystems Touch

To follow the main goal of this project, allow OutSystems users to develop and test their

applications continuously on a mobile device, the following architecture has been proposed.

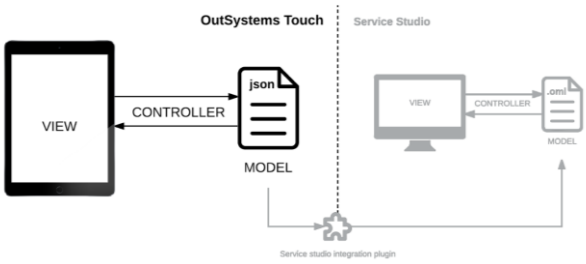

Figure 1 – Architecture

As depicted in Figure 1, the OutSystems platform IDE, called Service Studio, uses a Model-View-Controller (MVC) architecture that separates the implementation into three distinct layers and is a standard for interactive applications:

- Model - responsible for reading, writing and validating the application model, that contains a representation of its interface, business logic and data;
- View - responsible for displaying the application model to the user;
- Controller - responsible for processing the user's requests and changing the application model accordingly.

OutSystems Touch Scope

Service Studio allows users to develop complete web and mobile applications, as it allows the creation of processes, interfaces, business logic and the data model. The interface of Service Studio has tabs to develop each one of this parts. Due to time restrictions, the scope of this project, OutSystems Touch, is going to be the user experience associated with the development of interfaces directly on the final device. This means that OutSystems Touch only allows users to develop the application interface. Processes, logic and the data components are part of the future work as will be explained later on.

The architecture of the proposed solution also follows the MVP model, as OutSystems Touch was developed using Service Studio and the applications it generates follow this same model.

Figure 2 shows in more detail the architecture of the model of the OutSystems Touch, represented by a class diagram:

- An application contains screens.
- The screens contain widgets that are fragments of UI components.
- Widgets are buttons, texts, icons, etc. and define how the data is shown to end user (lists, graphs, images) with their properties.
- Properties can be colors, sizes, names, among others.

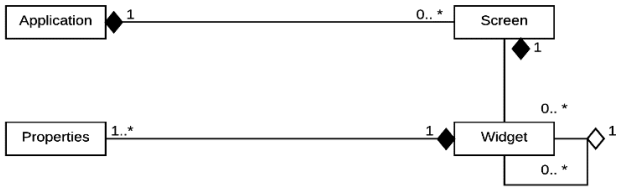

Figure 2 - Model of OutSystems Touch

4. Preliminary Work

To develop OutSystems Touch it was necessary to do work prior to the implementation. It started with user research, which counted on interviews to understand the users, defining personas and scenarios. Then it continued with learning the technology that would be used for the development of OutSystems Touch, OutSystems, as well as its evaluation with usability tests. This resulted in understanding the real needs of the users, leading the development to a familiar result from the existing version, with the possibility of improving usability and eliminating some of the more common current user errors. It was also defined the target application to be used later in all the usability tests and formative evaluations, in order to maintain coherence. After this, a small research about the touch challenges of the development of an application for touch environment was done. To design an app for a touchable environment that already exists on the desktop, it is necessary to maintain continuity, consistency and maintain brand (the look of each version should be similar) to convey familiarity with the existing system. However, there are several factors that make this process very challenging [2]:

Mobile screens are smaller. Mobile screens are physically smaller and the user can see a lot less information at once. Smaller screens means fewer pixels than desktop displays, too. As such, it is important to present important information "above the fold", use an easy-to-read font, and not overwhelm the user with too much content on the page because simple is best and less is more. So the use of tabs and trays is great to stretch screen space.

Slower processors. Mobile devices generally have much less processing power than desktop computers, so mobile apps can run slowly. This problem is already being solved with the arrival of the iPad Pro and Microsoft Surface.

Touch input. Mobile users work using their fingers and this has several implications, like no hover events or mouse pointers. On the existing OutSystems platform, these are used a lot. It is also an issue because the finger is thicker than the mouse pointer, so it is important use large controls and indicators that resemble their physical equivalents and use gestures to enhance the experience for your mobile users like swiping, pinching, and so on.

Inadequate keyboard. Normally, mobile devices have a little keyboard or tapping a minuscule on-screen keyboard, so typing on them is still far from being a pleasant experience. It is easier if there are autocomplete function to text fields and search fields, reducing the inputs required from users.

Mobile devices are becoming the prevailing computing platform. But the needs of mobile users are very different from those of desktop users. There are still very few developments or high productivity applications for mobile devices, or which are properly optimized for use in these devices. It is important to change this, following the evolution of the technology, providing a great work and development experience in a touchable environment. When the mouse appeared in 1968, existing software and users took time to adapt to this new, more natural, input device. The same seems to be happening with touch today.

After this, it was created a low-fidelity prototype as well as a formative evaluation and consequent iteration. The resulting prototype is presented in figure 3 and figure 4.

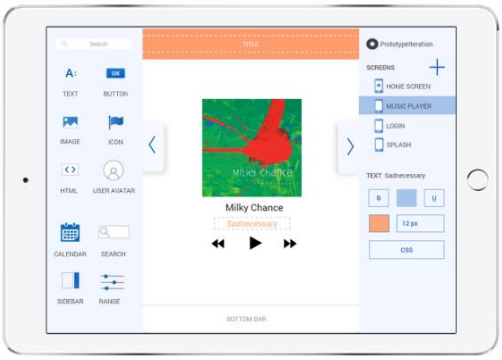

Figure 3 – Prototype screen when user is editing a text widget.

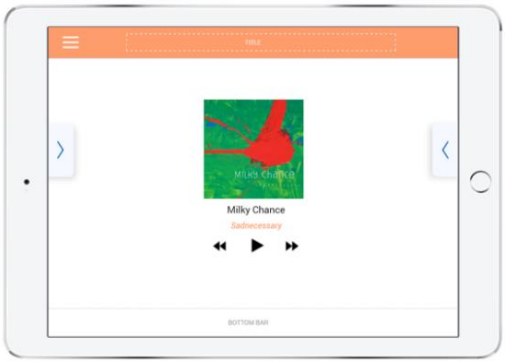

Figure 4 – Prototype screen when user is viewing an app.

5. Implementation

With the prototype described in the previous section, the implementation of the OutSystems Touch resulted in the definition of the necessary components and layers that allow the final user to develop a mobile application directly from the touch environment. In order to have a more clear picture, we started by defining what type of widgets and their properties was necessaries and the functionality to edit screens was added.

Then, we changed the structure of the application to be saved to a database. After this, we prepared a first functional prototype and then, we performed a formative evaluation to clarify doubts and to decide what was the best way to build a user-friendly application. We finally developed some additional functionalities to improve the prototype.

5.1. Creating Widgets

At this point we had target screens with a visual language that needs to be translated into computers language, or in this case, OutSystems language and be persisted between user sessions. So, we started by adding the widgets (Text, Button, Image, Icon and User Avatar) to the user interface that we wanted to implement.

Each widget was mapped to an OutSystems object and the status changes of each widget are Client Actions on the OutSystems Service Studio platform.

To add widgets, it was created the client action called *WidgetKindOnStarDrag* (figure 5) to be executed when a widget starts dragging. This client action allowed to create a widget with the default/initials properties and execute another client action, *DrawWidget, to* draw the widget depending on the kind and the corresponding default properties. It was decided to do two distincts client actions for this because as the properties of the elements are changeable, it will be necessary to use the *DrawWidget* more often.

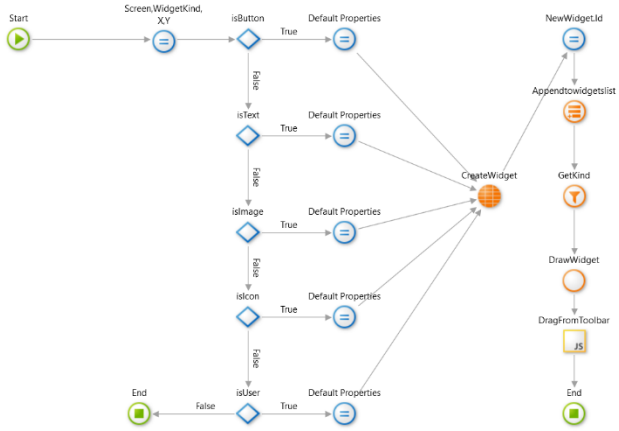

Figure 5 – *WidgetKindOnStartDrag* client action.

5.2 Allow Screen Edition

This is basically allowing the edition of the different widgets that are present in the screen which can be translated to the corresponding OutSystems actions. To allow screen edition, it was necessary to define several steps: define the selected widget and show the widgets' properties and edition of the widgets properties.

Define the selected widget and Show the widgets' properties

To define and show the selected widget by highlight was raised a new client action, called *OnWidgetSelect*. The widget structure created can be seen in figure 6. The properties of each widget are presented in structure.

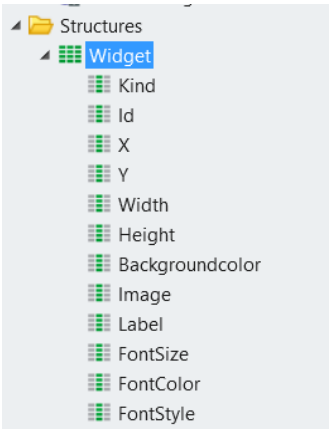

Figure 6 – Widget Structure on Service Studio.

Edition of the widgets properties

To edit the widgets properties' the client action *DrawWidget* was used, created to draw the widgets on the first drag with the initials properties. This was possible because properties are fields that allowing the user to input data. So, when the user changes the property field, the client action *ChangeProperties* is executed in order to draw the widget with new properties.

5.3 Storage

After allow screen edition, the next step was storage data. There are many advantages to store data in databases:

- Data security;
- On-Demand Scalability;
- High Performance;
- Minimized data inconsistency;
- Improved data access.

Therefore, a data model to storage data of the OutSystems Touch was created.

After this, we decided to add a menu sidebar like it appears in Service Studio when an application is created, and we also added an icon to the widgets sidebar to allow users to see the final application, hiding the two sidebars, opening in this way, the View mode. When this mode is active, it only appears on the screen an icon that allow to back to the Edit mode, showing again the both sidebars. At this stage, the first functional prototype was done. There were still many improvements to do. But first, the formative evaluation was done on the prototype that can be seen in figure 7 to help clarify doubts and make decisions, as explained in the next section.

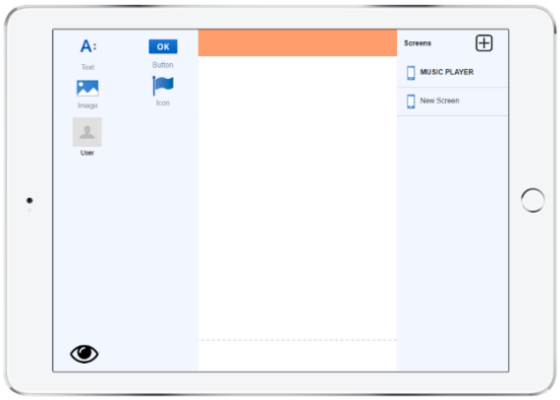

Figure 7 – First functional prototype.

5.2 Formative Evaluation and Prototype Iteration

As a result of the formative evaluation, it was decided that it was better to gather all the content just in a unique sidebar (with tabs, the first one to add widgets and the second with the properties), because they took up more screen space than desired.

To improve the prototype, and also as result of the formative evaluation, a work list was created to improve the user experience, eliminating the keyboard needs, taking into account the natural gestures of the touch environment and also to improve the functionality and the flow of the OutSystems Touch:

- Automatically move the tabs of the edit sidebar to "Properties" when a widget is selected
- Facilitate editing of text properties (bold, italic, normal, size)
- An application menu with links to all screens is automatically built
- Do not open the keyboard when using color picker or when change the icon selected
- Allow renaming screens
- Allow deleting screens and widgets
- Resize images with two fingers (pinch gesture)
- Lines to guide alignments of the elements
- Add link property to allow linking screens
- Suggest colors already used in the application, in addition to the color picker option
- Distinguish between the view mode or edit mode
- Allow scroll in edit sidebar when there are many screens and the sidebar is larger than the height of the device screen.

Developing and Testing

During the implementation of these improvements, we used the Chrome Simulator to test my code and used its debugger when necessary. Because of this, we came across a few unexpected things when, in the end, we tested OutSystems Touch on the final device, the iPad. For example, to rename the screens, the solution developed consisted of, after double click, to open a popup that allowed to rename the screen. This feature does not work on iPad, so we took another approach - the use of long press. Another example, it is not possible to simulate the pinch touch gesture in Chrome, so the development time of this functionality was much higher than expected, since it was necessary to always test on iPad and also because the iPad did not have debugger.

Ironically, these kinds of problems that arise from developing on a different device than the one used by end users, are exactly the ones that will no longer be a problem when applications are created with OutSystems Touch.

5.3 More Features

Applications Page

Service Studio allows users to build more than one application, so it was decided that the OutSystems Touch should allow the creation of more than one application simultaneously, too. So,

the data model for the OutSystems Touch was changed to support the new feature. The final data model of the OutSystems Touch can be seen in figure 8.

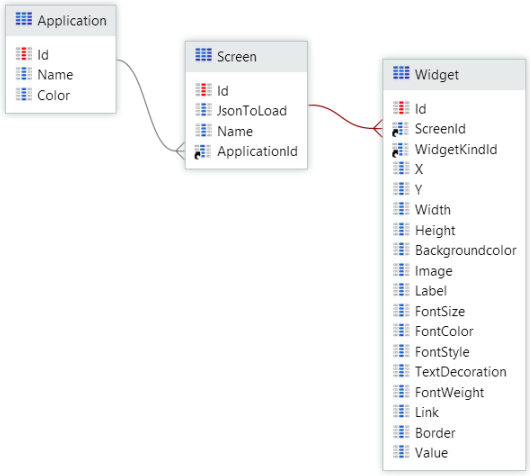

Figure 8 – OutSystems Touch data model.

Animations to improve the user experience

Well-designed animations make the difference in modern applications. Animations reduce cognitive load and prevents change blindness. Well thought-out and tested functional animation has the potential to fulfill multiple functions like visual feedback on server actions, navigational transactions, among others. This is a natural part of the design process and can turn a digital product with memorable good experiences and will cause users to instinctively like an app.

So, we decided to add animations to edit sidebar. The edit sidebar moves to the right and to the left, as we know. Sometimes, jumping users to a new position without transition is confusing. So, we used the animation to smoothly transport users between the positions of the edit sidebar. This way, we can guide the user's attention in ways that both inform and delight. These animations are also used on the tabs that allow users move to add widgets or see or edit widgets properties and are also used to hide edit sidebar when users clicked on view mode or to show it when users clicked to open edit mode.

Adjustments of interface details

In the end, we decided to make some adjustments to OutSystems Touch to make this application more familiar to current Service Studio users, like using Service Studio icons and adding the color of the Service Studio to the edit sidebar. In figures 9 and 10, we can see Service Studio and OutSystems Touch interfaces to compare both.

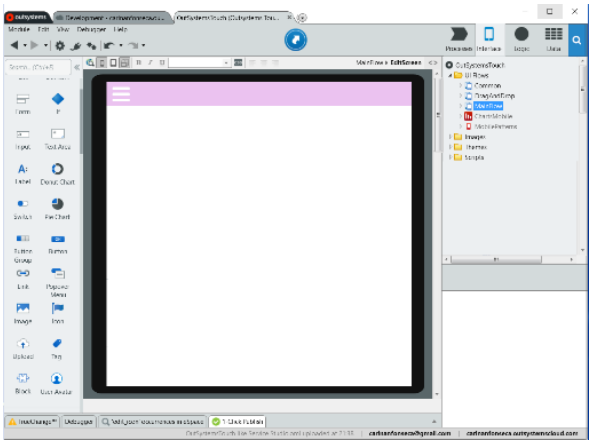

Figure 9 – Service Studio interface to build interface of applications

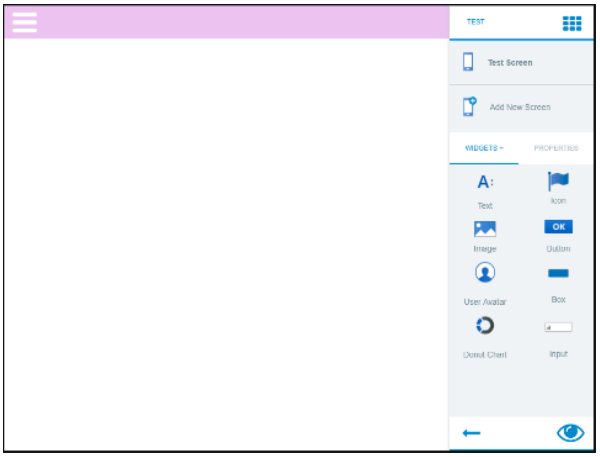

Figure 10 – OutSystems Touch interface to build interface of applications.

6. Evaluation

The evaluation allows us to assess the effectiveness, efficiency and usability of a system and therefore it is essential in this study.

Its main goal is to ensure that users can successfully complete the assigned tasks. It is important to perform a significant number of tests in order to get enough feedback and ideas for improvement.

The evaluation methodology of OutSystems Touch was based on the interactive

design method - performing usability tests based on low-fidelity prototypes that would be used to quickly test, iterate and refine the prototype until it reached good levels of usability.

The usability tests were performed with potential users: students of the Computer, Telecommunication or Electrical Engineering degrees.

The script for usability tests to present the testers consists in building a music player screen (target screen) and a menu application with link for the created screens.

The metrics analyzed were the following:

System Usability Scale (SUS) is a method of ascertaining the effectiveness, efficiency and satisfaction of a system. To measure it, the user answers 10 questions (this is the final questionnaire), with a scale from 1 (Strongly Disagree) to 5 (Strongly Agree). Then, it's necessary to do some calculations:

- For each of the odd numbered questions, subtract 1 from the score.
- For each of the even numbered questions, subtract their value from 5.
- Take these new values, and add up the total score. Then multiply this by 2.5.

If the result is

- 80.3 or higher, the system is considered an A: people love the system and will recommend it to their friends,
- 68 or thereabouts is a C, the system is doing ok but could improve, and if
- 51 or under is an F, which means it's important to make usability the top priority now and fix this fast.

The ranking can be seen in figure 11, too.

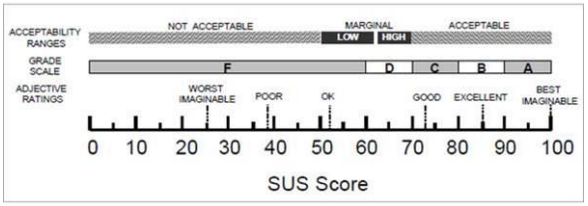

Figure 11 – Ranking of SUS

Task Time is a measure of efficiency and productivity. It records how long the user takes to complete a task in seconds or minutes, starting when user finishes to read the task scenarios and ending when the user has finished the whole task, including reviewing.

Task Completion consists of registering if the user finished the task successfully (1) or not (0).

Number of Errors consists of registering any unintended action, slip, mistake or omission a user makes while attempting a task, with a description and then we can add severity ratings to errors, or classify them into categories. Expectation evaluates how difficult users expect a task to be comparing it to actual task difficult rating (from the same or different users).

These metrics are the same used to performed the usability tests to Service Studio, in order to make a better comparison between the two systems.

The usability tests were performed on OutSystems Touch with potential users with the same way that the usability tests performed to Service Studio. They were composed of a small introduction for contextualization, task accomplishment, final questionnaire to later calculate the System Usability Scale and a debriefing.

6.1 Results

To discuss the results, we did a box plots to show distribution of a set of data, comparing the results of the analyzed metrics (task completion, number of errors, time and SUS) to the tasks of the usability tests performed to both systems. Here, we presented the box plots relative of the SUS of both systems:

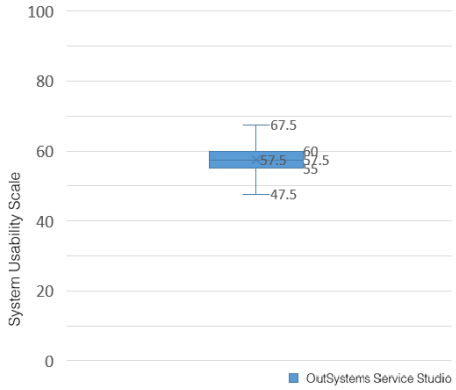

Figure 12 – Box plot of the usability scale of OutSystems Service Studio.

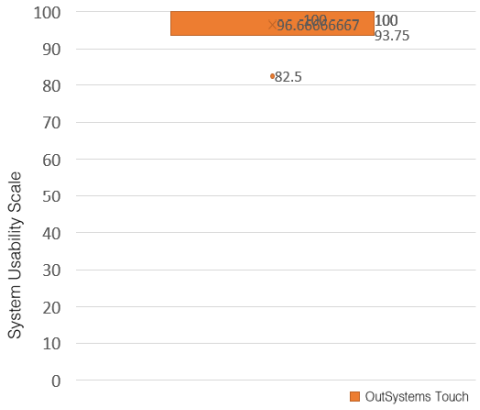

Figure 13 – Box plot of the usability scale of OutSystems Touch.

When OutSystems Touch is used, a higher average in usability scale is recorded (97) than OutSystems Service Studio (58). Service Studio was evaluated with a C in System Usability Scale, which means that the system is acceptable but could be improved and OutSystems Touch was evaluated with an A in System Usability Scale, which means that the people love the system and will recommend it to their friends.

At the end, we can conclude that with OutSystems Touch it was possible to perform the same tasks more efficiently and efficiently, with more satisfaction than in OutSystems Service Studio.

We can conclude that with the final prototype of the OutSystems Touch it was possible to perform the same tasks more efficiently and efficiently, with more satisfaction than in OutSystems Service Studio, because it has higher scores in the metrics defined than the current version of the platform, objectively bringing OutSystems closer to the goals defined in the introduction:

1. New users find it easier to learn how to use OutSystems for front-end development.
2. The development experience is more suited for devices that allow touch as well as typing, such as Microsoft Surface.
3. The prototype makes is easier to perform quick development and test cycles, directly on the device.
4. Developers are able to quickly tweak their applications, in the field and in close contact with the end user.

7. Conclusions

This article presents a prototype of an iPad version for the Service Studio IDE, called OutSystems Touch. We began by contextualizing the OutSystems platform and concluding that there is still nothing for building software in a touch environment in the enterprise market. We also analyzed the challenges of creating mobile versions of existing desktop apps, so that we are aware of the problems that are usually faced when using complex apps in a mobile environment.

The research and learning of this OutSystems technology was conducted to better understand the real needs of the users and to familiarize ourselves with the platform. To complete this challenge, we evaluate the current state of this platform and collect sufficient data to compare with OutSystems Touch in the end, with the possibility of improving usability.

After this, we designed the prototype of OutSystems Touch taking into account the touch challenges previously analyzed and we started the implementation. We did three iterations of the prototype after other three evaluations.

Finally, we present the final evaluation of the OutSystems Touch where we concluded that it has a very positive evaluation within the stipulated timeframe and the proposed goals.

It's believed that the development of this IDE will contribute to the breakthrough of touch technology, by delivering a prototype of the first fully visual low-code platform for mobile devices, directed to the enterprise market. Most importantly, OutSystems Touch allows OutSystems users to develop and test their apps continuously on a mobile device through a Service Studio that is easier to learn, can be used anytime, anywhere, and brings developers, end-users and mobile devices closer together.

References

[1] C. Richardson and J. R. Rymer, "Low-code development platforms: The 14 providers that matter most and how they stack up," The Forrester, April 2016.

[2] N. Golas, "User interface design best practices," March 2014.