Evaluation of the Development Framework Jasonette

Nicolas Schmitt[1], Tim Wesemeyer[2], Marcel Kissel[3], Dennis Horvat[4], Marco Wünschel[5], Michael Sattel[6], Yves Lehmann[7], Axel Herbstreith[8] and Michael Gröschel[9,*]

[1,2,3,4,5,6,7,8,9] Mannheim University of Applied Sciences, Germany

[*] *Corresponding author's email: m.groeschel [AT] hs-mannheim.de*

ABSTRACT— ***In this paper we evaluate the development framework Jasonette which is based on JSON. The evaluation was carried out by means of a prototype of an app informing prospective students about the study program Enterprise Computing at the Mannheim University of Applied Sciences. We defined a set of criteria that we cross checked during and after the implementation of the app.***

Keywords— Jasonette, Development Framework, Evaluation, Prototype

1. INTRODUCTION

The fundamental task of this project, which is assigned to the topic of mobile business, is the performance evaluation of the developer framework *Jasonette* [1]. This framework can be used to create native apps. These apps are converted from files whose content only consists of *JavaScript Object Notation* (JSON) format. To assess the usability and strength of this framework, a prototype app has to be created with this tool. This app is designed for Android and iOS and is about the study program Enterprise Computing at the Mannheim University of Applied Sciences [2]. In the following we use the abbreviation 'UIB' for the study program which refers to the German name '**U**nternehmens**i**nformatik' with a **B**achelor of Science degree.

The idea of the app was refined during the project progression. Meanwhile the target group has been defined to those who are interested in studying UIB. App users should be able to find out whether Enterprise Computing is in general the right subject for them. For this case, there is a suitability test in the app. The app should help these people to find out if studying that subject at the Mannheim University of Applied Sciences is suitable for them. Some of the already existent content from the website of the Mannheim University of Applied Sciences should be included in the app, e.g. the description of the course. Furthermore, the app should help prospective students to find their way at the Mannheim University of Applied Sciences. Ideas for the design came, among other things, from personal experiences and problems the developers of the project had at the beginning of their studies. For example, semester dates and the campus plan are easily accessible in the app. We also want to promote the study program and motivate people to study UIB in Mannheim. Therefore, we have created a video in which we asked students why they are studying at our university and what they like about UIB. This app should really improve the process by which people find out if this study program is the right one for them. There is even a chatbot planned. This chatbot should provide answers to most questions interested people ask, and thus relieve some professors of answering frequently asked questions.

As already described the aim of the project is the assessment of Jasonette. We want to know if this framework is as effective as it is advertised, due to its simple and easy-to-learn usability, or if there are restrictions and limitations to be accepted. There are many reasons why we like Jasonette but there are also aspects we don't like about it. All benefits and detriments we found are described compactly and precisely on the following pages.

2. RELATED WORK

In today's world, where apps have become increasingly important to society, it is essential that apps can be designed quickly and easily [3, 4]. In the meantime, there are many app kits with the common goal of generating own apps as quickly as possible without higher technical effort and specific knowledge [5]. At the turn of the millennium, the first app-building kits came on the market. These differ in many criteria, e.g. the user-friendliness, features, compatibility of the devices and the support of the operators [6, 7].

These app kits address users without programming skills. Using such app kits, for example, owners of small companies can design and maintain simple apps. The possibilities are limited to the offer of the respective app kit. Usually, it is not possible to meet the need for individual elements.

On the other hand, there is a range of frameworks, many of which are available for free and released under an open source license, which are more flexible and have more options in terms of customization of the apps. However, this is at the expense of greater demands on the competences of the developers. Most frameworks use a combination of HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) and a scripting language like JavaScript or TypeScript. On this basis, a native app is then generated. The focus is always on the approach of keeping the cross-platform code platform as broad as possible, which largely applies. However, details are to be implemented dependent on the respective target platform. A popular representative is React Native [8], by which also the popular Facebook app is implemented for several mobile operating systems. A number of representatives is based on the framework Apache Cordova [9], such as Adobe PhoneGap [10]. Further known representatives are Nest [11], Weex [12], Ionic [13] based on Angular [14] or NativeScript [15], compared in [16]. It has to be taken into account that all comparisons only represent a snapshot in time at a certain point in time and that changes may occur very rapidly over time. Individual assessments based on a list of criteria are suitable for a specific evaluation [17, 18].

Although some well-known frameworks for cross-platform development have a declarative approach, the idea and implementation of Jasonette is unique in this conclusiveness and – as far as can be seen – this is not followed by any other framework in the same way.

3. APPROACH

The basic approach of evaluating the Jasonette framework was to create a mobile application prototype for prospective UIB students at the Mannheim University of Applied Sciences. We approached the problem by doing secondary research in order to gain a basic understanding of Jasonette and its functions. Afterwards, due to the lack of statistics and figures for Jasonette, we decided to evaluate the framework from an end-user perspective by implementing the above-mentioned prototype. A set of criteria was established serving as foundation for the evaluation process. The content of the app to be implemented was determined by researching information provided by the Mannheim University of Applied Sciences via its official website, flyers and videos. Topics to be covered in the app were selected and prioritized using the technique of open card sorting. The students' own experiences and problems during their time as prospective students or applicants were considered as well. Drafts – first on paper, later with the software Axure – helped to design the layout of the app. The implementation of the prototype was done in Eclipse IDE, PHPStorm and AndroidStudio. The implementation was realized with the help of the Instagram template provided by Jasonette. During the implementation and by its completion we could identify the strengths and weaknesses of Jasonette by means of the predefined criteria.

4. GENERAL EVALUATION CRITERIA

This chapter describes a selection of criteria for evaluating development frameworks for mobile applications in general. These evaluation criteria provide the basis for the following evaluation of Jasonette.

4.1 Familiarization Effort

The workload of familiarizing oneself with a framework is an essential criterion. The relation between the benefits of a framework and the time to become acquainted with it determine how widely used and accepted it becomes. If as much time is needed to read the documentation and understand the framework as it is needed to implement something 'manually' without using the framework, it is useless to apply it.

4.2 Documentation

The documentation of a development framework is a major part for the success of the framework. Using any tool, real success starts by understanding why its use is helpful. Concerning this, a clear structure and precise explanation of individual contents is very important. If the documentation is too complex and confusing, many people will not give the framework a chance, even if it might be more effective than other frameworks. Therefore, a good documentation should contain implementation examples, code explanations and tutorials. With the help of visualizations, introduction videos and pre-built templates the user's interest can be aroused to give the framework a try. The documentation shall help developers to easily install the required components and quickly gain a basic understanding of the framework. For advanced users it is important that the available components are documented in detail and that there is a good overview of all implementation possibilities.

4.3 Community (Forum, Slack)

The size of the community supporting a framework is another important aspect. A user forum can be useful for getting help and support concerning certain unsolved problems or errors for no documentation is all-embracing. Exchange about updates, functionality or (potential) use cases of a framework is another important advantage coming along with a supporting community.

4.4 Cross-platform

A program is platform-independent (or cross-platform) when it can be used on different platforms without special adjustments for different operating systems or hardware specifications. Applications typically require a runtime environment where they are installed and can run stably. In case of a platform-independent application different hardware and software constellations can be used, and it doesn't matter which operating system is installed. The main advantage of cross-platform development is that the code doesn't have to be adapted to different platforms and is only written once. On the one hand, the implementation effort will be reduced. On the other hand, it simplifies the maintenance of the code.

4.5 Pre-built Templates

Pre-built templates represent a major advantage if offered by a framework. They serve as example for implementations and can be starting point for own adjustments, so save time and workload on condition that these templates are easy to comprehend and that there is an adequate documentation offered.

4.6 Graphical Design Possibilities

Graphical design possibilities or just called styling offers possibilities to design the user interface. The native development offers many possibilities. Nevertheless, there are frameworks that further enhance styling or those that simplify the implementation of styling. In simplifying the efforts often limits must be accepted.

4.7 Integration of Additional Frameworks

Using frameworks, the range of functions should be expanded, or the programming effort should be reduced. In general, it can be said that a framework should facilitate the development of software. Frameworks develop their full potential when they can be combined. The question arises how easily this can be realized, and whether a different framework can be integrated directly into Jasonette.

4.8 Debugging

It is important that users of the framework have a way to easily find errors in their code within an editor or a development environment. Especially with regards to a more extensive code, the error search can quickly become very complex and time-intensive. Therefore, a development framework should offer an integrated debugger to find errors quickly in the program logic. In this case, it should be possible to control the program sequences by setting breakpoints to see where errors occur. Furthermore, precise error messages must be available in a console, for example. If there is no way to search for errors, it can happen that some errors can only be found after a long search or even not at all.

5 PROTOTYPE UIB-APP

Within the evaluation of the development framework Jasonette, we developed a prototype to examine the possibilities and limits of the framework. The evaluation criteria described in the previous chapter were considered in the implementation process to analyze if Jasonette fulfills the qualities of a good development framework. The development of the prototype of the UIB-App and the examination of the criteria created the basis for the evaluation of Jasonette. The app is intended to provide first-hand information and impressions about the study program Enterprise Computing at the Mannheim University of Applied Sciences. With regard to this potential, applicants can get information about the campus or important content of the study program, in addition upcoming appointments are listed. In this chapter, the functionality of the prototype is described in detail and the realization of individual criteria is explained using the prototype.

In the beginning of the prototype development, we first familiarized ourselves with the framework using the official documentation of Jasonette. The documentation provides a step-by-step guide on how to use the framework and a detailed description of the various components. The explanations include many examples, which is helpful to get a basic introduction into the framework. In the first phase of development the question arose which operating system should be used with regard to the app, and if a cross-platform development exists concerning Jasonette. Basically, Jasonette offers a cross-platform development but only for the best known operating systems Android and iOS. Referring to this, the framework provides a container app in which metadata can be stored to define a name, description or icons. The metadata also includes an URL, which refers to the file where the JSON code of the app is stored. For the implementation of the structures, contents and functions of Jasonette, only the description language JSON is needed. Hereby it is not necessary to learn a new programming language. In order to build the app and create the installation files for Android and iOS we had to install the corresponding development environments (Android Studio for Android and Xcode for the Apple platform).

Although we followed the steps explained in the documentation to create the app for iOS, it was not possible to build or run the prototype for this operating system. In summary, the development with Jasonette is only partly platform independent since there is currently no implementation option for alternative operating systems such as BlackBerry OS or Windows Phone. During the implementation of the prototype, we partly used pre-built templates provided by

Jasonette. Regarding this, the official documentation offers the JSON code for various layouts or features. However, the code structure for a Jasonette app is always the same. Figure 1 shows this structure based on the start screen of the prototype of the UIB-App. Referring to this, the app is divided into three areas: header, body and footer. Within the header the title of the app is displayed, and different menus can be inserted by the developer to let the user navigate through the app. Through the footer, it is possible to switch between different screens and see the content of the app, which is stored in the body. Our example uses German for the entire content of the app because our target audience lives in the area around the university.

Figure 1: Start screen of the app

In relation to the header area and the presentation of the title of the app, there were first restrictions in the implementation. Looking at the title design (located in the header), first constraints appear: On the one hand, it is not possible to insert the title along with an icon but to display either the title or the image. On the other hand, it is mandatory to align the logo centrally – interspaces need to be adjusted via image processing. Furthermore, it is not possible to create a link for the icon to reference internal sites of the app (although it is stated as possible in the corresponding documentation). Besides the representation of static text contents, the framework offers the option to integrate MP4 videos or embed YouTube videos into the body of the app. For our prototype, we inserted a self-created information video into the start screen of the app. By means of the footer, it is possible to navigate to different app sections and gather information about the studies, application requirements and upcoming deadlines. The performance of several pages is quite bad due to delays while loading content.

In the section ‘Bewerbung’ (application), prospective students can do an aptitude test, in order to get a first impression of the study program and to get a feedback if they are qualified or not (figure 3). In this test, the prospective students can answer the questions by choosing checkboxes. In the Jasonette framework radio buttons and checkboxes are not included. That is the reason why we used a self-written PHP script which is included as an iframe. In the section ‘Termine’ (dates of term) (figure 2), we tested if it is possible to link the Jasonette app to a database. The shown dates of term are written in a MySQL database, read by a PHP script and displayed in the app.

Figure 2: Display of the dates of term

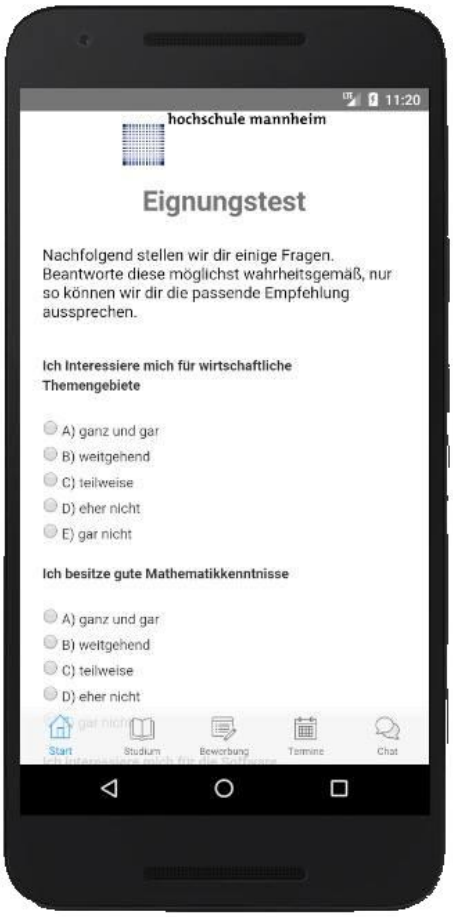

Figure 3: Integrated aptitude test

When implementing the prototype, please note, that some components and functions described in the documentation are not really working. There were some difficulties when errors occurred because it is only possible to check the JSON syntax with an editor. There are no integrated possibilities to search for errors, so it is not possible to set breakpoints, for example, to get closer to the logic to investigate. To avoid errors, it is advisable to do minimal changes in the JSON code and to test the app right thereafter. Jasonette does not provide any error messages if the JSON code is incorrect. Furthermore, implementation limits have been introduced in some places, so that the frameworks have their own components which do not allow modifications. Other possibilities must be found using workarounds.

6. EVALUATION OF JASONETTE

In this section, the criteria described in the section above will be applied to Jasonette. The criteria are examined regarding the described prototype.

6.1 Familiarization Effort

Jasonette advertises with its low familiarization effort: “With Jasonette, you don’t even have to be a programmer. Anyone can make an app. And do it in minutes, not days or weeks.” [1]

Trying to implement an app with Jasonette without having programming experience may work, but not within minutes. A user does not know how to set up a development environment without ever having written software. In fact, first results are accomplished within one hour, maybe less, if the user is not a beginner. The point is that this app developed in minutes does contain some data displayed on the smartphone but it hardly contains any application logic. To implement a more complex app with a larger functionality range, Jasonette claims more time and effort. Expertise in coding enormously promotes the quality and logical capabilities of the app, as well as the ability to integrate external contents. To summarize, only one of the statements quoted above proves true. Either, the user is no programmer and therefore does need more time than just a few minutes to create an app, or the user can do it in minutes, but he is a programmer with experiences in JSON, development environments, HTML and CSS, at least.

6.2 Documentation

Jasonette provides a clear and detailed documentation, including a brief introduction to developing with this framework. With the help of the step-by-step instructions for installing and using Jasonette, the user can set up his first

own project relatively quickly. The documentation describes the use of Jasonette for Android and iOS. It provides detailed information about available components, layouts or templates, for both, newcomers and advanced users. Furthermore, it offers tips for programming with Jasonette. The explanations are illustrated with pictures, videos and sample codes, which make it easy to understand and to learn about the framework in a short period of time. A negative aspect, with regard to the documentation, is that some explanations do not work as described. For example, it is not possible to insert a separate menu icon into the header, although all steps are performed as described in the documentation.

6.3 Community (Forum, Slack)

Jasonette references a community forum on its website, used by about 150 users. The topics are divided into four categories: Help, General, Showcase and Uncategorized. With 14 topics per month, Help is the most used category, which emphasizes the main function of the forum – to offer support for Jasonette implementations. Another way of communicating with the Jasonette community is joining on Slack. More than 600 users are registered there, for example, on a random Monday in the project phase, around 20 users were online. To summarize, support by community is given, but there should be more users to cover a wider area of topics and problems.

6.4 Cross-platform

The platform independence is fulfilled only in part with Jasonette. Although the code is written in JSON and therefore not tied to a specific programming language or operating system, currently it is only possible to set up the app for Android and iOS. This makes it possible to develop applications for the two most popular and most frequently used operating systems, but no reference is made to alternative operating systems such as Windows Phone or BlackBerry OS.

6.5 Pre-built Templates

Jasonette offers two kinds of templates. First, it provides code sections to be used and adjusted, for example, for JavaScript or several JSON code abbreviations. This represents more a tutorial with example code than templates. Second, there are complete ‘examples’ available ready for use, like Jasonpedia, a layout adjusted for online encyclopedias, or a Twitter UI, also completely written in JSON. These files comprise around 200 lines of code. They help to understand the functioning of Jasonette and illustrate the functional range of the framework. For the realization of the prototype, the example Instagram UI was used as basic layout and starting point for the implementation, which is another positive aspect – the reuse of existing templates.

6.6 Graphical Design Possibilities

With Jasonette it should be easy to create native apps which are only using JSON files. The question is, how powerful this tool is in relation to the design of the graphical user interface. We wanted to know if there are any restrictions in exchange with the easy-to-learn usability of Jasonette.

During the implementation of the app, it has turned out that one is very limited in his possibilities. It is not possible to have access to any native design elements and there are too few templates. In the native Android app development, there are tab layouts which are very useful. As already mentioned, such helpful templates are missing. This means, that there are no checkboxes or radio buttons available.

Another problem we found is that it is not possible to change the size or position of a picture that should be displayed in the header. The picture is displayed at the center of the header. Alternatively, HTML could be imported to style the app by inline styling. The problem is that this solution isn’t elegant and doesn’t give any advantage. Furthermore, it isn’t possible to link another page when clicking on the icon in the header section. In the Jasonette documentation it says that this should be possible, but it isn’t yet. It should also be possible to open another Jasonette page by clicking on the menu point in the upper right, but it doesn’t work.

6.7 Integration of Additional Frameworks

The question was raised whether other frameworks can be integrated into the development framework Jasonette to extend the functionality. For example, a framework could be used for testing the functionality of the source code. There are many frameworks out there for testing. For instance, there is the module testing framework Jasmine [19] which can be used for JavaScript code.

As we wanted to know if there are any frameworks which are combinable with Jasonette, we found out that it is not possible to integrate any framework into it. There is only one possibility to combine another framework with Jasonette: The code has to be developed externally with another framework and an interface, which can be used by Jasonette, has to be offered (for example a REST interface). As a result, the options and possibilities are limited. The problem is that even this workaround isn’t easy to realize, and it is not very effective. The reason for this is that one is restricted by processing the delivered data in Jasonette. This causes a lot of problems. The better way would be programming the app with native code and integrating another framework into it. This saves a lot of extra effort.

6.8 Debugging

Currently Jasonette doesn't provide a debugger to search the code for errors. Using an editor, only the syntax of JSON will be checked for correctness. However, it won't be checked if the code is correct regarding the specifications of Jasonette. Furthermore, it's not possible to use debugging methods like breakpoints to examine individual code blocks for their logical correctness and occurring errors. If an error occurs, there is no error message displayed in a console, for example, therefore it is unclear why the code doesn't work as expected. Own output logs can only be inserted at particular spots in the code in order to localize the error. To avoid errors, there shouldn't be made too many changes in the code at once. To keep the error range small, after small changes implementations should be executed. According to contributions in the Jasonette forum, a debugger is currently in development.

We also had a problem with iOS and Android compiling. First, we focused on the Android app and observed that the design fits. As we tried to compile the code to an iOS app, the design looked completely different and it was not usable. This means that the code must be adjusted for iOS. It entails a big effort to port the app for iOS. It doesn't make sense for a platform-independent framework that the code must be adjusted for every platform.

6.9 Protection and Security

We want to add one more aspect for the evaluation concerning the protection of your intellectual property contained in your code. In terms of IT security, as a developer, it is important to hide confidential data. Otherwise, it would be easy for people to obtain this information. Therefore, we wanted to know if there are any possibilities or guidelines in Jasonette to hide information. Using only Jasonette, it is not possible to hide data. This is because Jasonette, as the name already suggests, is based on JSON, which was primarily not intended to hide information. JSON is a standard for exchanging data in an independent format. Everything that is put into a Jasonette code can be seen in the app, if only Jasonette is used. The only possibility to hide data is, for example, to build a backend with a server-side programming language like PHP. Confidential information can be hidden there and only wanted information is delivered to Jasonette. Jasonette represents the data in the frontend.

7. CONCLUSION

The idea behind Jasonette is powerful. To build an app according to the modular principle, only with a JSON structure, seems to be simple. It is not a great effort to achieve first results with the framework because no programming knowledge is necessary. Especially apps with static content can be implemented quickly and easily. As soon as data is to be loaded from a backend, it becomes more complicated. By missing syntax check and error messages, the development gets unnecessarily complicated. Here, results can be achieved faster at native development level. The announced compatibility on the platforms Android and iOS could not be confirmed during the evaluation of the framework. While the app works well under Android, it is useless for iOS. At this point, further improvements are necessary. Considering that the framework is in an early stage of development, we can ignore some errors. The idea behind Jasonette is awesome, but still in their infancy (only a single release, version 0.2.0, February 2017). We can look forward to how the developer of the framework continues the project.

8. REFERENCES

[1] Jasonette – Native App over HTTP. (n.d.). Retrieved February 19, 2018, from https://jasonette.com/
[2] Study program Enterprise Computing at the Mannheim University of Applied Sciences. (n.d.). Retrieved February 19, 2018, from https://www.english.hs-mannheim.de/study-programmes/bachelor-courses/enterprise-computing.html
[3] Blüml, A., & Frank, A. (2011). Wie viel kostet die Entwicklung von Apps? Grundlagen – Ablauf – Stundensätze – Musterkalkulationen. Norderstedt: Books on Demand
[4] Apps for everyone. (2012). Entrepreneur, p. 46
[5] Höpfner, H., Pencke, J., Wiesner, D., & Schirmer, M. (2011). Towards a target platform independent specification and generation of information system apps. ACM SIGSOFT Software Engineering Notes, 36 (4), 1-5. doi: http://dx.doi.org/10.1145/1988997.1989010
[6] Gröschel, M., Vo, S., & Kunzmann, J. (2013). Baukästen zur Realisierung von Mobile Apps: Potenziale, Bewertung, Marktanalyse. horizonte, 42, 26-30
[7] Gröschel, M. Mobile Apps: Alternative App-Baukästen? digitalbusiness CLOUD 05/2013, 44-46
[8] React Native. (n.d.). Retrieved April 3, 2018, from https://facebook.github.io/react-native/
[9] Apache Cordova. (n.d.). Retrieved April 3, 2018, from https://cordova.apache.org/
[10] Adobe PhoneGap (n.d.), Retrieved April 3, 2018, https://phonegap.com/
[11] Nest (n.d.), Retrieved April 3, 2018, http://www.nestpia.com/
[12] Weex (n.d.), Retrieved April 3, 2018, https://weex.apache.org/
[13] Ionic (n.d.), Retrieved April 3, 2018, https://ionicframework.com/
[14] Angular (n.d.), Retrieved April 3, 2018, https://angular.io/
[15] NativeScript (n.d.), Retrieved April 3, 2018, https://www.nativescript.org/

[16] Idesis, S. (2017). Retrieved April 3, 2018, https://www.outsystems.com/blog/free-cross-platform-mobile-app-development-tools-compared-2017.html
[17] Palmieri, M. Singh, I., & Cicchetti, A. (2012). Comparison of cross-platform mobile development tools, 16th International Conference on Intelligence in Next Generation Networks, Berlin, 2012, pp. 179-186. doi: 10.1109/ICIN.2012.6376023
[18] Dalmasso, I., Datta, S. K., Bonnet, C., & Nikaein, N. (2013). Survey, comparison and evaluation of cross platform mobile application development tools, 9th International Wireless Communications and Mobile Computing Conference (IWCMC), Sardinia, 2013, pp. 323-328. doi: 10.1109/IWCMC.2013.6583580
[19] Jasmine Documentation. (n.d.). Retrieved February 19, 2018, from https://jasmine.github.io/