# What is the impact of various front-end web development frameworks to the energy consumption of mobile web applications?

Davide Mariotti
2640152
d.mariotti@student.vu.nl

Luca Grillo
2640151
l.grillo@student.vu.nl

Marios Evangelos Kanakis
2619105
m.e.kanakis@student.vu.nl

Anastasios Sidiropoulos
2631597
a.sidiropoulos@student.vu.nl

***Abstract*—*Context*. The need for cross-platform coverage and reducing development times is apparent in the web development industry, and that is demonstrated by the abundance of web front-end frameworks that have been introduced to the market in the recent years. Comparing these frameworks is far from an easy task, and current studies have focused on aspects such as lines of code or size. In this study, frameworks are compared in regards to their energy consumption, CPU usage, and memory consumption.**

***Goal*. The goal of the study is to provide guidance to developers, so that they are able to make better informed decisions when needing to choose among the vast amount of web front-end frameworks that exist today.**

***Method*. An empirical study was conducted on an identical application made using 9 different web front-end frameworks. We measured each of our dependent variables (energy consumption, CPU usage, and memory consumption), under 20 trials, where an independent variable, the scenario (an automated set of pre-defined actions), was executed in each trial ensuring a standard execution of the experiment across all occasions.**

***Results*. The results show that there are significant differences in energy consumption, CPU usage, and memory consumption. It is also apparent, that Clojurescript is the worst in the context of our study, as it is the least energy efficient, with the highest CPU usage, and a fairly high memory consumption. Additionally, we provide Dojo2 as the best option, as results show that it achieves an adequate balance between the three variables.**

***Conclusions*. There are significant differences in all three of our variables. However, even though the least optimal framework has been identified, and some guidance has been given pertaining to the framework that satisfies all three of our measured variables, the general idea is that certain frameworks do more well than others depending on what aspect is used to judge.**

## I. Introduction

In the modern age of information and communication systems, traditional desktop applications have moved to the cloud. The need for applications to be cross-platform and offer mobile device support, has gotten the user interface to lie in its entirety through the window of a web browser [1]. This rapid growth and expansion of mobile devices has pushed software developers to explore, evaluate and experiment with several web front-end frameworks necessary for mobile web applications development. Mankovski has tried to provide a guideline on how to find the "best" web front-end framework [2].

During the recent years, an abundance of front-end web development frameworks have been introduced to the market. As web application development can become a tedious process, when having to tweak the web application front-end to have the desired look on mobiles in addition to personal computers. Around 60% of software developers try to combine two or more different web frameworks in order to provide users with the desired look and feel on mobile devices[2]. Each framework offers a variety of UI components and libraries aimed to make the development process easier. On a scale from 0 to 5, developers rated their satisfaction as 3.8, when asked about the current state of web front-end frameworks [3]. Apart from the look and feel, developers want to know which web front-end frameworks are optimized and offer great performance. Developers are constantly looking for web front-end frameworks that consume less battery in users devices.

As mobile software applications operate in resource-constrained environments, guidelines to build energy efficient applications are of utmost importance [4]. The aim of our experiment is to evaluate the energy consumption of each framework, using different measurements that will be taken when an Android device is running an web application. The results could reveal the frameworks with the highest energy consumption, granting developers additional guidance for taking informed decisions. The focus of our experiment will be on a project called "RealWorld" created by Eric Simons [5]. Each implementation of this project uses the same HTML structure, CSS, and API specifications, but a different library/framework [6]. The "RealWorld" project is available in the frameworks displayed in Table I.

In the last column of the table it is possible to check which frameworks we are going to use in our experiment. With ClojureScript + re-frame and Elm we had problems when we tried to run the frameworks. Instead, with Aurelia, AppRun and Crizmas MVC we could not run the frameworks on our mobile due some problems with the npm packages.

While the web front-end development frameworks that we will be testing are used on a multitude of projects everyday for example famous web application like PayPal and Netfix make extensive use of AngularJS [7] or Instagram are slowly inserting react into their code [8], not much research has been conducted to understand their differences in terms of energy

TABLE I: Frameworks tested.

| Framework | Initial Release | Last Release | Official Website | Involved Languages | Developers | Involved in our experiment |
|---|---|---|---|---|---|---|
| React/Redux | 2 June 2015 | 4.0.0 / 15 April 2018 | redux.js.org | JS | Dan Abramov and Andrew Clark | ✓ |
| Angular | 14 September 2016 | 6.1.7 / 6 September 2018 | angular.io | TypeScript | Google | ✓ |
| Elm | 2012 | 0.19 / 21 August 2018 | elm-lang.org/ | HTML / CSS / JS | Evan Czaplicki | X |
| Vue | February 2014 | 2.5.17 / 1 August 2018 | vuejs.org | JS | Evan You | ✓ |
| React/MobX | 31 August 2015 | 5.1.2 / 10 September 2018 | mobx.js.org | State Managment | Andy Kogut | ✓ |
| AngularJS | 20 October 2010 | 1.7.4 / 7 September 2018 | angularjs.org | JS | Google | ✓ |
| Aurelia | 10 December 2014 | 1.3.0 / 2 July 2018 | aurelia.io | JS | Aurelia | X |
| Svelte/Sapper | 11 December 2017 | 0.21.1 / 24 September 2018 | sapper.svelte.technology | JS / HTML | Svelte | ✓ |
| ClojureScript + re-frame ⋆ | 13 October 2011 | 1.10.339 / 25 June 2018 | clojurescript.org | JS | Rich Hickey | X |
| Angular + ngrx + nx ⋆ ⋆ | 14 September 2016 | 6.1.7 / 6 September 2018 | angular.io | TypeScript | Google | ✓ |
| AppRun | 7 October 2016 | 1.15.2 / 3 August 2018 | apprun.js.org | Elm | Yiyi Sun | X |
| ClojureScript + Keechma ⋆ | 13 October 2011 | 1.10.339 / 25 June 2018 | clojurescript.org | JS | Rich Hickey | ✓ |
| Dojo2 | March 2005 | 3.0 / 24 July 2018 | dojo.io | JS | Dojo Foundation | ✓ |
| Crizmas MVC | 2018 | 1.0.2 / April 2018 | npmjs.com/package/crizmas-mvc | React | Raul-Sebastian MihĂČilĂČ | X |

⋆ Information for ClojureScript.
⋆ ⋆ Information for Angular

efficiency. Moreover, computational demands are increasing, therefore draining mobile devices' battery at a quicker pace. For developers, it is of great importance to have knowledge of the frameworks that perform tasks faster, so they can provide maximal user experience, while preserving the user's battery.

Thus, the purpose of this experiment, is to fill the gap in energy efficient research regarding mobile devices and more specifically, for the various web front-end frameworks that are necessary for web applications development. In addition, we seek to explore and compare the differences in performance with energy efficiency, if there are any.

The paper is organized in seven parts as follows. First, the experiment is defined stating the Goals, Questions and Metrics in Section II. By doing so, the main aim of the project is easily understood. In addition, the questions and metrics will allow us to reveal our results. The experimental planning in Section III explains in more details the context, scope, and variables of the experiment. In this section, a hypothesis is formulated, based on which, the experiment is executed. A detailed description of the experiment execution can be found in Section IV where the structure of the experiment and the data analysis techniques are thoroughly explained. In section V the statistics and results of the experiment are presented. Following, in section VI you can find discussion about the interpretation of our work related to the experiment. Threats to Validity are discussed in section VII. Finally, section VIII concerns the main conclusions of the experiment.

## II. Experiment Definition

The Experiment Definition will be given in terms of the GQM method illustrated by Basili [9]. This method has an hierarchical structure based on three different levels.

### *Goal*

The objects studied are 9 web front-end frameworks. The purpose is to evaluate these different frameworks. The perspective is from the developer‘s point of view, in the context of mobile web development. We define the goal of our experiment with the template provided by Basili [9]. The goal is summarized as:

Analyze **web front-end frameworks**
for the purpose of **evaluation**
with respect to their **energy efficiency**
from the point of view of **software developers**
in the context of **mobile web development**.

### *Questions*

We are going to form questions based on the goals that we defined in the section above:

**RQ1** *What is the impact of web front-end frameworks on the energy consumption of mobile web applications?*

To answer this question, we will examine the "RealWorld" example application regarding whether or not there are differences in web front-end frameworks concerning energy consumption. More specifically, we will evaluate if web front-end frameworks play a significant role towards energy consumption and how big of an impact they have, if any.

*H0: There are no significant differences regarding energy consumption and web front-end frameworks.*

**RQ2** *What is the impact of a front-end web development framework on the performance of mobile web applications?*

In this question, we seek to explore if the usage of different front-end web development frameworks is related to performance. To answer the question, we will measure how the performance of the related framework corresponds to the power consumption measurements of the examined web application. Additionally, this will indicate whether or not a specific front-end framework is an optimal choice regarding performance and power consumption, respectively.

*H0: There are no significant differences between web front-end frameworks and performance.*

**RQ2.1** *What is the impact of a front-end web development frameworks on memory consumption?*

In order to elaborate on the question posed at RQ2 it is important to ask how each web front-end framework is performing in computational terms. To answer this question, we will measure the memory consumption of each web front-end framework when executing a specified scenario. The differences between the various web front-end frameworks will give us a good indicator on the performance of mobile web applications.

*H0: There is no significant difference in memory consumption between web front-end frameworks.*

**RQ2.2** *What is the impact of a front-end web development frameworks on CPU usage?*

To answer this question, we will measure the CPU usage of each web front-end framework when executing a specified scenario. The differences between the various web front-end frameworks will give us a good indicator on the performance of mobile web applications.

*H0: There is no significant difference in CPU usage between web front-end frameworks.*

### *Metrics*

In this section we proceed to associate the questions with the appropriate metrics. The defined metrics are data that are gathered throughout the execution of our experiment. The metrics are used for the statistical analysis and are very significant for proving or rejecting the initial hypothesis formed for each question. For this experiment, the metrics associated with the formulated questions and hypotheses are specified as follows:

**Power consumption:** The power consumed by executing and performing a specified scenario in the "RealWorld" example application. Measured in mW (milliWatts).

**Scenario completion time:** The time required to complete a defined set of actions in the "RealWorld" example application. Scenario completion time is measured in milliseconds.

**Energy consumption:** Energy consumption is derived by multiplying power consumption and scenario completion time. Energy consumption is measured in Joule.

**Memory consumption:** Memory consumption is defined as the memory consumed during the execution of the web application. Memory consumption is measured in kiloBytes (kB).

**CPU usage:** The level of CPU usage during the execution of the scenario. The unit of the metric is percentage.

### *GQM Tree*

In Figure 1 the GQM Tree is shown. The tree is based on the goal of the proposed experiment on the conceptual level, which is connected to the formulated questions regarding the operational level and tied to the metrics as the quantitative level as specified above.

## III. Experiment Planning

In this section we will explain how the experiment will be conducted. The goal definition was presented in the last section, now we discuss some important points for the planning of our experiment. First of all, the environment in which the experiment will be performed is described, discussion about variable selection will follow. Hypotheses are formulated and the population sample of our experiment is defined as well. In the last part of this section we present the experiment design and instrumentation that we will use in the experiment.

### *Context Selection*

The context of the experiment is defined based on four dimensions, as described by Wohlin et al. [10]:

- Off-line vs. On-line
- Student vs. Professional
- Toy vs. Real problems
- Specific vs. General

The experiment will be conducted in an offline situation, in the same room and with the same internet connection. The applications that we are going to evaluate are already built and we are not involved in the development process. Instead, we will create a scenario for each application, that offline simulates the behavior of the average user. So, no real users will be using the application during the experiment.

In regards to the second dimension, 'Student vs professional, the experiment is for professionals, since the treatments are applied by professional developers to the objects, which are the web front-end frameworks.

In terms of toy vs. real problems, the experiment concerns a real problem even though it is a test application due to the fact that it was developed by real developers and not ourselves. Furthermore, it is a real problem because it will attempt to find out how various web front-end frameworks vary in regards to power consumption and performance. Thus, the outcome of the experiment might be used to provide guidelines to software developers as to which web front-end framework is the most energy efficient.

Finally, as mentioned above, the experiment serves a general context due to the fact that some of the most popular and widely used web front-end frameworks will be examined [3].

### *Variable Selection*

Choosing the independent and the dependent variables is a crucial step that must be done before the start of the experiment. For this experiment, three independent variables are used. The first one is the web front-end frameworks used for web applications development and the second one is the scenario, which is a set of predefined actions that will be performed on the Chrome browser of the provided Android device. Both independent variables are categorical.

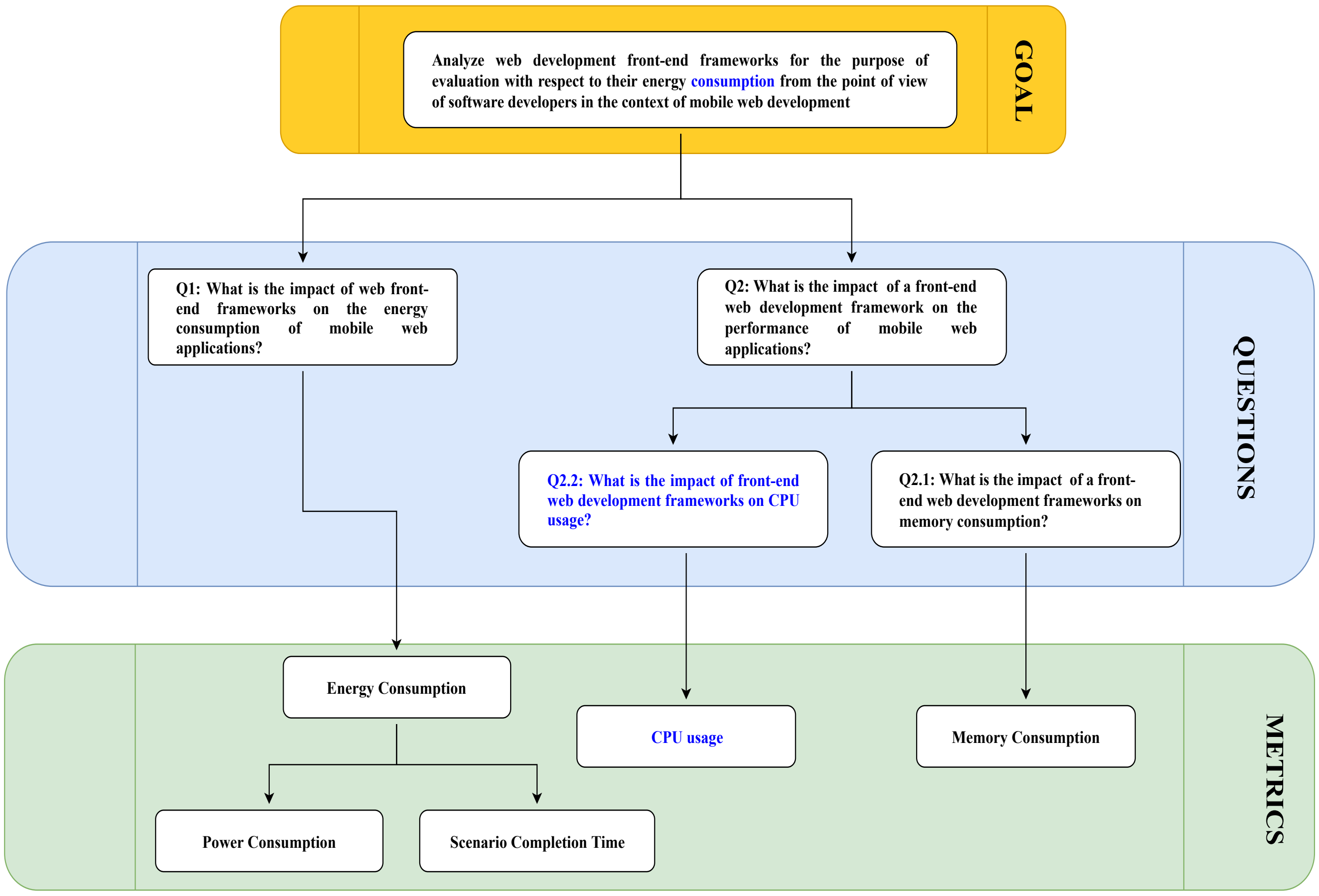

GOAL
Analyze web development front-end frameworks for the purpose of evaluation with respect to their energy consumption from the point of view of software developers in the context of mobile web development
QUESTIONS
Q1: What is the impact of web front-end frameworks on the energy consumption of mobile web applications?
Q2: What is the impact of a front-end web development framework on the performance of mobile web applications?
Q2.2: What is the impact of front-end web development frameworks on CPU usage?
Q2.1: What is the impact of a front-end web development frameworks on memory consumption?
METRICS
Energy Consumption
CPU usage
Memory Consumption
Power Consumption
Scenario Completion Time

Fig. 1: The GQM tree of the experiment

The web front-end frameworks variable will include nine treatments. As described already and displayed in Table I, each treatment corresponds to a different web front-end framework which was used to build the same web application.

For the scenario variable, we are going to apply only one treatment. The treatment is defined as loading up the web application, login with a predefined user account and publish a post. It is clear that two different scenarios could produce different results. For example *react/redux* could be better than *angularJS* in a simple scenario (just load the home-page and scroll it) but worse in a more complex scenario (like login into the application and publish an article). In order to not overload our experiment we choose to focus just on one more complex scenario. Thus, it can be considered as fixed factor.

The third independent variable is the internet connection. The internet connection will be used to connect the web front-end framework to the applications' back-end which runs online. We will execute the experiments in the same room and in the same conditions, in order to have the same internet connection for the entirety of the experiment.

On the other hand, there are five identified dependent variables for this experiment. All dependent variables derive from the Goal Questions and Metrics approached as described in Section II. Also, all our dependent variables are numerical.

Furthermore, Memory consumption is measured in kiloBytes (kB). The Memory consumption identifies the required memory needed to successfully run the web application depending on the web front-end framework and the scenario.

CPU usage is measured in percentage. The percentage of CPU usage indicates how much of the processor's capacity is currently in use by the system.

Scenario execution time is the third dependent variable. This variable describes the required time to complete a scenario with a specific web front-end framework measured in seconds (s).

As a fourth variable we identify power consumption. Power consumption is measured in $\mu$Watts. In our experiment, it is defined as the total power consumed through the execution of a scenario with a specific web front-end framework.

Energy consumption is the last dependent variable. This variable is measured in Joule (J) and is a product of power consumption in Watts (W) and scenario execution time in seconds (S). Energy consumption will be calculated for each experiment run for each scenario and each web front-end framework.

## *Hypothesis Formulation*

This section is where our hypotheses to be tested are formally defined. We provide a null and alternative hypothesis for each research question. For the statistical tests, we use a confidence level of 95 percent.

For RQ1, which aims to evaluate the impact of web front-end frameworks on energy consumption of mobile web applications, we obtain measurements from the "RealWorld" application, for each of the 9 different frameworks. Then, we formalize the null hypothesis, that there is no difference in the energy consumed between web front-end frameworks.

$H_0 : \mu_{JF_1} = \mu_{JF_2} = ... = \mu_{JF_9}$

Where $\mu$ denotes the mean, J represents the energy consumption in Joule and $F_{1...9}$ represents the specific web front-end framework for each framework tested from Table I.

And the alternative hypothesis $H_1$:

$H_1 : \exists\ \mu_{JF_x} \neq \mu_{JF_y}$

Where x and y represent two specific web front-end frameworks given from the the frameworks tested from Table I.

For RQ2, we attempt to evaluate the performance of each different web front-end framework, and for that we split the research question into two parts, the first being measuring memory consumption and the other CPU usage. We formulate null and alternative hypotheses for each part.

We formulate the following null hypothesis for memory consumption:

$H_0 : \mu_{MEMF_1} = \mu_{MEMF_2} = ... = \mu_{MEMF_9}$

Where $\mu$ denotes the mean, MEM represents memory consumption in kiloBytes (kB) and $F_{1...9}$ represents the specific web front-end framework for each framework tested from Table I.

And the alternative hypothesis $H_1$:

$H_1 : \exists\ \mu_{MEMF_x} \neq \mu_{MEMF_y}$

Where x and y represent two specific web front-end frameworks given from the the frameworks tested from Table I.

Regarding CPU usage, we formulated the following null hypothesis:

$H_0 : \mu_{CPUF_1} = \mu_{CPUF_2} = ... = \mu_{CPUF_9}$

Where $\mu$ denotes the mean, CPU represents CPU usage in percentage and $F_{1...9}$ represents the specific web front-end framework for each framework tested from Table I.

And the alternative hypothesis $H_1$:

$H_1 : \exists\ \mu_{CPUF_x} \neq \mu_{CPUF_y}$

Where x and y represent two specific web front-end frameworks given from the the frameworks tested from Table I.

## *Subject Selection*

The definition of population and sample is very important when conducting an experiment [11]. In order to examine energy consumption of different web front-end frameworks, the experiment is based on a population represented by all the web front-end frameworks that exist. It is easy to understand that a huge number of web front-end frameworks exist. As the population is unknown we applied *Non-Probability sampling*.

A *Convenience sampling* technique is used to select our sample from the population. This technique allows us to choose most convenient web front-end frameworks from the population as subjects for our experiment. Table I shows the 9 frameworks chosen as sample. These 9 web front-end frameworks all implement the same application, so we can easily compare each framework.

However, there is another factor that needs to be considered. Each version of the application, in terms of the web front-end framework used, is developed by different developers. On one hand this means that we have a better and wider sample since our experiment aims to provide guidelines for all developers. On the other hand, we might face a negative impact due to the fact that different developers have different coding styles and not all write the same quality of code in terms of efficiency and performance.

## *Experiment Design*

In this section a thorough description will be given regarding the design of the experiment. Experiment design concerns both the organization and the execution aspect. The choice of an accurate design will have an impact on the statistical analysis and ultimately, the results of the experiment.

The object of the experiment is the RealWorld example application. All treatments will be applied to the same web application. This will help us provide accurate results regarding the impact of our factor.

For the experiment, we have identified one main factor with 9 different treatments. The factor is web front-end frameworks and the various treatments are the frameworks.

In addition, the *scenario* is regarded as a fixed factor. As already specified a *scenario* is a predefined set of actions that will be executed in the web application. This fixed factor can be something that might affect our results, but our main focus does not concern this aspect.

Moreover, internet connection will be considered as a fixed factor. The same internet connection will be used throughout our experiment. We are aware for unstable internet connection or not reliable thus, we will try to minimize the effect the internet connection will have on our experiment by using the same connection, in the same room and network.

The experiment is identified as having one main factor with multiple treatments. Thus, a *Randomized complete block design* technique will be applied in order to assign each treatment to our subject. Additionally, for more accurate results we will take repeated measurements of the experiment. For each web-front framework the experiment will be executed 20 times to acquire more reliable data. Additionally, the amount of measurements is going to be consistent in all treatments in order to achieve a balanced design.

## *Instrumentation*

In order to achieve the goals of our experiment, a set of tools/instruments will be used. The specified tools and instruments will help us monitor, execute the application and collect the necessary data for our analysis.

**R and RStudio**
R is an open-source programming language commonly used for data analysis and statistical tests, which R allows for easy execution and quick integration.
RStudio is an open-source IDE(Integrated Development Environment) that provides an interpreter for the R statistical language that we will be using to write code that runs statistical tests and plots data.

**adb**
Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with an Android device.

**Trepn Profiler**
Trepn Profiler is an on-target power and performance profiling application for mobile devices. It gives you an insight into things such as a real-time view of individual CPU cores and a view of network usage (cellular and Wi-Fi). Additionally, it enables monitoring of GPU frequency and displays battery power in mW. Finally, it offers an advanced mode where users can select data points and save data for later analysis.

**Monkey Runner**
The monkeyrunner tool provides an API for writing programs that control an Android device or emulator from outside of the Android code. With monkeyrunner, you can write a Python scrip that installs an Android application or an existing package, which executes it, whilst taking "screenshots" of its user interface, and stores them on the workstation. Thanks to Monkey Runner's API we will able to reproduce the scenario we have described above.

**Android Task runner**
Android Task runner is a Python framework for automating experiments on Android. Thanks to this framework we will be able to execute our experiment.

**Android device**
Huawei Nexus 6P is the device on which we run our experiment. Android version is 6.0.1. The device is clean and without any element that can influence the experiment.

**MacBook Air**
For the experiment we will use a MacBook Air where all the different web front-end frameworks that the web application has been developed with will be executed. The OS of the MackBook that we will use is macOS High Sierra, version 10.13.6. The processor is an 1.3 GHz Intel Core i5 and it has 4 GB 1600 MHz DDR3 of memory.

**Chrome**
Chrome will be used as the default web browser of the provided Android device.

## IV. Experiment Execution

**Preparation**
The preparation stage includes a number of different settings that need to be well set up and finely configured before the actual execution of the experiment. Using the AndroidRunner framework it enables us to specify the environment of our experiment. Furthermore, via the configuration settings we have to provide AndroidRunner with the web browser that the experiment will be executed upon, the necessary profilers, namely MonkeyRunner and Trepn, as well as the data that AndroidRunner will collect through the profilers during the execution.

Additionally, we will have to provide the scenario of the execution in terms of steps that will be conducted. In our setting, the scenario is defined as follows:
*The first step is to open the web browser and load the web application. Afterwards, a "user" logs in and from the menu the "Create New Article" button will be clicked. Then, the "user" will write an article, filling the text fields, and finally publish it.*
In the scenario described above some options will remain static to increase the validity of our measurements. The first option is the "user" login process. The login phase for all experiments will include an already manually registered account to the web application at the user authentication process. Also, the article that will be written will be manually generated from us and will be the same across all experiments.

Furthermore, before the execution of the experiment we have to secure a stable WiFi internet connection that will be used across all experiments. The experiments will be run in the same room, laptop and internet connection throughout the whole execution stage to secure valid data collection.

In addition, the Android device that will be used for the experiment must be plugged to the computer at all times, be fully charged and at highest brightness setting. Since the same application will test throughout the whole experiment, the rendered pixels on the screen will be the same. Thus, the brightness setting can remain stable and at high values as it will not affect the energy consumption.

Another important preparation step is to ensure that the Chrome browser that is going to be used is clean. This will be achieved by running an "adb clear" command from the laptop. This means that it has its cache memory deleted as well as any browser history, data or files either generated or stored in memory. This step is to ensure that the execution environment of the web application remains identical across all runs of the experiment.

As a final preparation step, we have to set up all the web front-end frameworks running through the computer localhost with online access to the back-end of the application that the experiments will be executed upon.

**Execution**

After everything is successfully set up the execution of the experiment begins by running AndroidRunner, with the use of Python interpreter, and passing the configuration file as input. Based on the provided configuration file AndroidRunner will execute the scenario and collect all the data from the specified profilers. Every execution of the experiment (one for each frameworks) is repeated 20 times with 2 minutes of interval between each repetition. This amount of time is introduced to be sure to not influence tentative each other. A high-level overview of our execution environment is displayed in Fig. 2. In step 1, the computer via AndroidRunner commands the Android device to open the RealWorld example web application. Then, in step 2 and 3, the application sends an HTTP request through the computer which responds by serving the web page. Following in steps 4, 5 and 6 the web application requests data from the back-end through the computer, the computer takes the request and forwards it to the online hosted back-end of the web application before forwarding again, in step 7, the data back to the Android device to be rendered. This process will be invoked numerous times getting requests and responses back and forth until the specified scenario is completely executed.

When the execution of the scenario is over, the data collected will be stored in files of csv type. This data can also be aggregated through a python script to better match the settings of the experiment and make the statistical analysis easier.

Moreover, after successfully running the experiment for several times and ensuring that the data collected are correct and valid throughout all our applied treatments, namely the various web front-end frameworks, we will continue our experiment by using RStudio to analyse our data.

At this stage, we will load up the various data and start exploring them by plotting basic graphs, calculating the mean, median, minimum and maximum values. With these descriptive statistics we can check for outliers or maybe for plots that are heavily skewed and for unclean data. Accordingly, we will also check to find the distribution and see if there is a trend and/or whether the data falls under the normal distribution, which is important for simple statistics test such as the student test.

Afterwards, we will start applying statistics tests aiming to reject our initially specified hypotheses. Since we have one factor with multiple treatments our initial test is Analysis of Variance (ANOVA). With ANOVA we can start testing our hypotheses and see whether or not our treatments have significant impact, if any, on the subject. If ANOVA fails to provide accurate and reliable results we can perform the test 9 times with 1 factor and 2 treatments design and correct the p-values using the Ohlm correction procedure. Then, we will attempt to verify our results and provide strong evidence on our analysis.

## V. Results

In this section the results of our experiment are presented. There are two sub-sections. In the first sub-section, we will take a look at some descriptive statistics regarding the data collected. Then, in the second sub-section, we will provide in detail our hypotheses testing, based on the ANOVA tests that were conducted.

**Descriptive Statistics**

To start our statistical analysis the first step was to get some insight on our data. With some basic descriptive statistics we can have a first look on our collected measurements. The summary of our data is displayed in Table 2.

TABLE II: Data Summary

| CPU load | Memory Consumption | Energy consumption |
|---|---|---|
| Min : 15.69 | Min : 2641235 | Min : 7.512 |
| 1st Qu. : 17.95 | 1st Qu. : 2708368 | 1st Qu. : 21.518 |
| Median : 18.26 | Median : 2745440 | Median : 23.512 |
| Mean : 18.37 | Mean : 2737521 | Mean : 23.390 |
| 3rd Qu. : 18.63 | 3rd Qu. : 2771577 | 3rd Qu. : 25.527 |
| Max : 20.76 | Max : 2799732 | Max : 31.501 |

Unfortunately, from this summary we cannot conclude much. The only notable conclusion that can be made is that the Memory usage (kB) and Energy Consumption (Joule) are more spread out than CPU load. However, we should take into account that mobile devices nowadays have 3,4 or even 8 GB of Memory, and this statement based on the context may be translated that there is not a big spread in Memory usage (if we consider only 100.000kB difference).

After examining our data we decided to make graphical presentations that are suitable for providing us with valuable insight on our variables (CPU load, memory usage and energy consumption) and our treatments (web front-end frameworks). Here we present only the graphical representations that are meaningful. In Fig. 3 we see a boxplot based on the Subjects (-treatments -web front-end frameworks) and the energy consumption.

From this boxplot we can only assume that there are indeed differences between energy consumption and the various web front-end frameworks. Moreover, we can assume that some frameworks are worse than others. For example

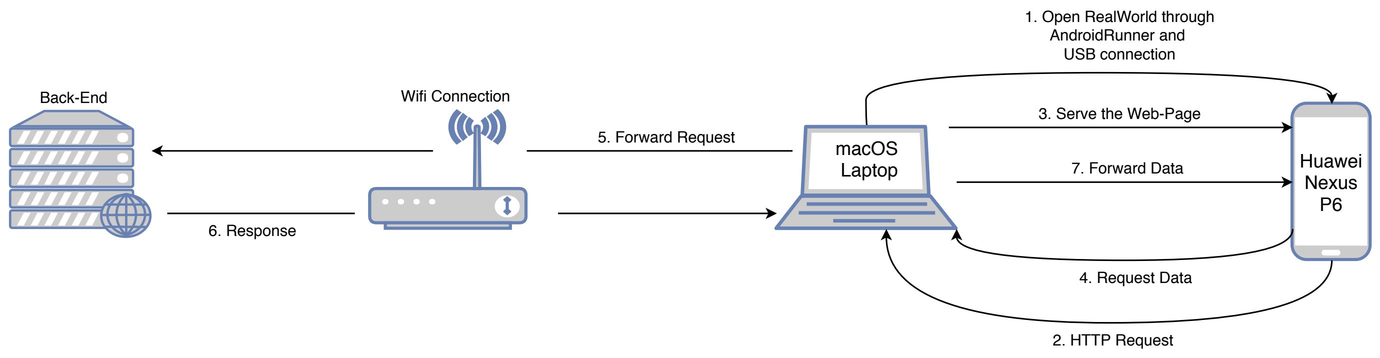


Fig. 2: Execution environment

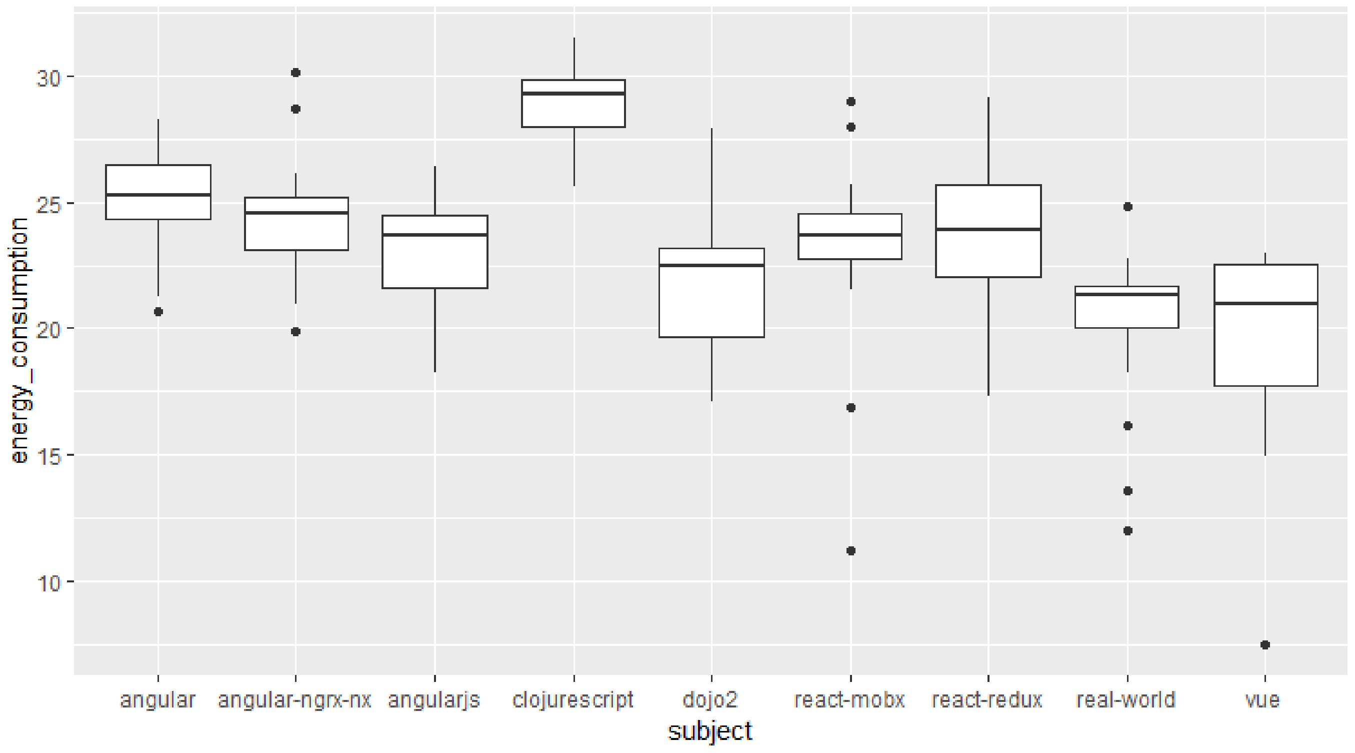


Fig. 3: Boxplot on Energy consumption

ClojureScript-Keechma seems to be the worst one regarding energy consumption. Following, in Fig. 4 and 5 we present another two boxplots, this time with CPU load and memory usage with respect to the different subjects (-web front-end frameworks). As with the energy consumption boxplot we can safely assume that for both CPU load and memory usage the use of different web front-end frameworks has an effect on those variables.

### Hypotheses testing

As discussed in Section IV in order to test our hypotheses we are initially conducting One-Way Analysis of Variance (ANOVA). Our factor is the "subject" mentioned above which includes 9 treatments, namely, the 9 web front-end frameworks in Table I. For each of our specified hypotheses in Section III we will be conducting ANOVA experiments to test our hypotheses and try to find strong evidence that proves research questions.

**Assumptions**
Before conducting the hypotheses testing we have to address the three requirements (-assumptions) that ANOVA poses. As described here [12], the following are the requirements for conducting ANOVA testing:

- The population of which the samples are drawn is normal
- Independence of cases, the samples must be independent from each other
- Homogeneity of variance

To test for the first requirement, the one of normality,

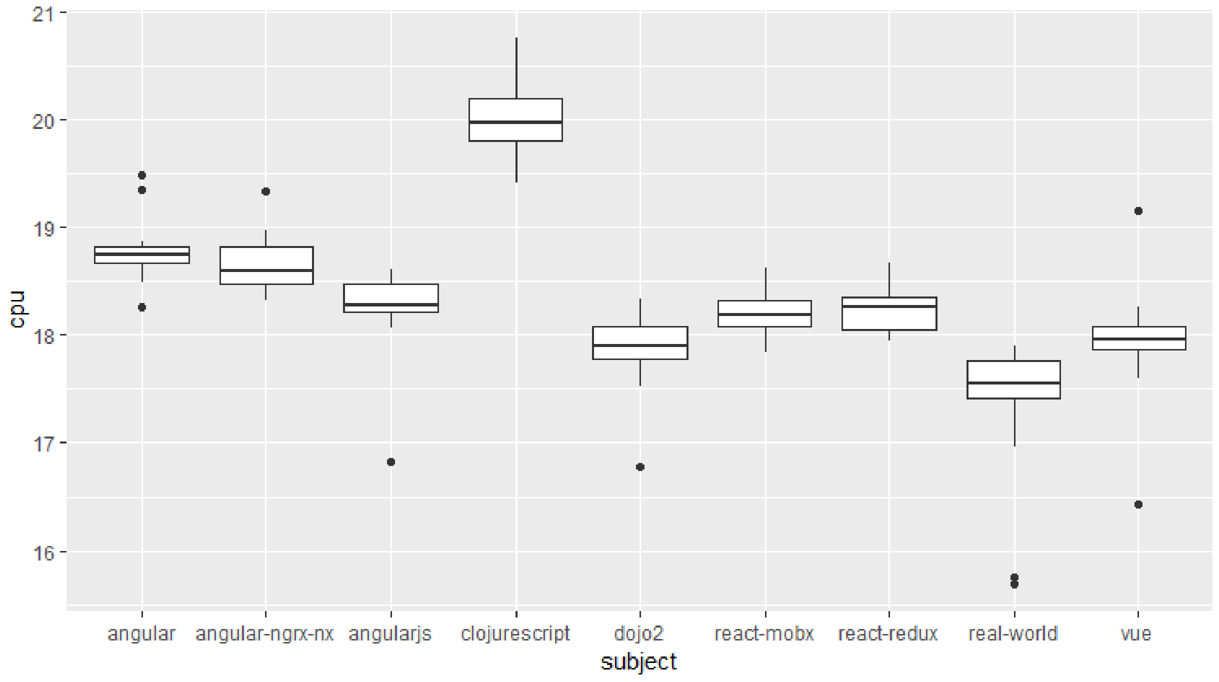


Fig. 4: Boxplot on CPU load

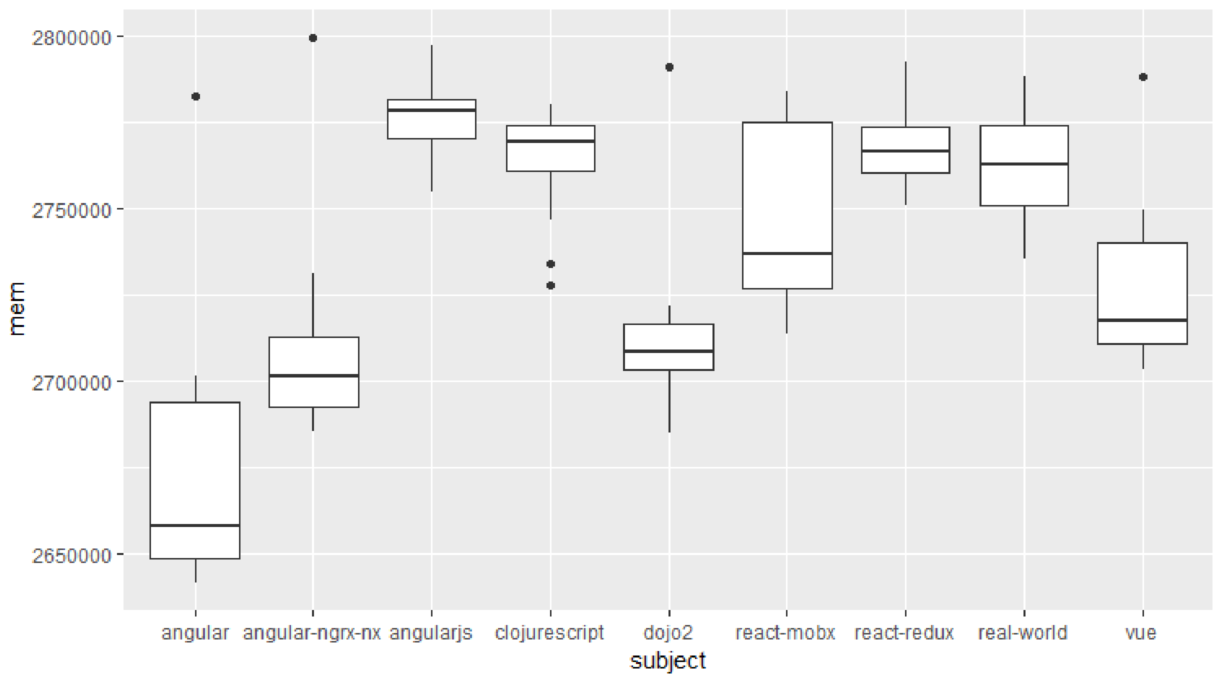


Fig. 5: Boxplot on memory load

we draw the corresponding qq-plots. As displayed in Fig. 6 normality cannot be assumed. Additionally, there are a few outliers and that makes it more unclear whether or not our sample is drawn from a normal distribution. Therefore, a Shapiro test for normality was conducted for each of the variables with an alpha significance level of 0.05. Namely, the p-value for the energy consumption is **0.0001**, the p-value for the CPU load is **4.568e-08** and lastly, the p-value for the memory usage is **3.317e-07**. This means that we cannot assume normality for our sample and hence, the first requirement for ANOVA is rejected.

For the second requirement, based on the setting of our experiment we have achieved independence of cases.

Finally, for homogeneity of variance we conducted Levene's test. The p-value for the energy consumption variable is **0.7096**. For CPU load the corresponding p-value is **0.5453** and for memory usage the p-value found on Levene's test is **0.004849** which is lower than the alpha significance of 0.05. As can be seen here, only memory usage fails to meet the homogeneity of variance requirement. As can be seen, the first requirement is not met and for the third one memory usage does not have homogeneity of variance. This may indicate that the use of ANOVA as our hypotheses testing test may not be a good idea. However, as stated in [12], ANOVA can still be used and be robust. The only case where ANOVA cannot be reliable is when the independence of cases is not achieved. Thus, we decided to use ANOVA but not take the outcome as granted. In order to provide further evidence and build a stronger case, we also conducted a Kruskal-Wallis test which is a non-parametric alternative for the ANOVA test. If ANOVA and Kruskal-Wallis agree on the testing of our hypotheses. Then, we have strong evidence and can be sure about the results.

**ANOVA**
We conducted three separate One-Way ANOVA tests, based on the three formulated hypothesis described in Section III. The results are aggregated in Table. III.

TABLE III: ANOVA

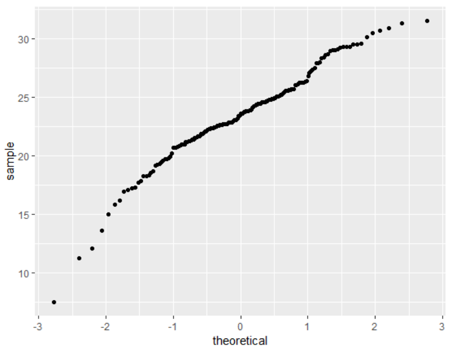
| | Mean Sq | F value | Pr(>F) |
|---|---|---|---|
| Energy Consumption | 147.572 | 18.345 | < 2.2e-16*** |
| CPU Load | 10.8947 | 83.575 | < 2.2e-16*** |
| MEM Usage | 2.3906e+10 | 53.079 | < 2.2e-16*** |

As it appears, web front-end frameworks do have significant difference on all three variables (energy consumption, CPU load and memory usage). Moreover, by providing the summary of the ANOVA we can see which web front-end frameworks have the most significant difference on energy consumption, CPU load and memory usage. This is displayed in Table. IV.

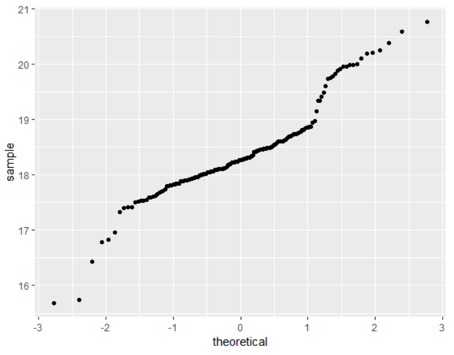
Furthermore, we conducted a pairwise t-test for all subjects (-web front-end frameworks) and their observed variables to identify which pairs of frameworks have significant differences among them. The results are indicated in Table. VII, Table. VI and Table. V.

**Kruskal-Wallis**
To provide further evidence of our findings and validate ANOVA tests results, we proceeded to conduct a Kruskal-Wallis non-parametric test, which does not assume normality or homogeneity of variance. The results are displayed in Table. VIII.

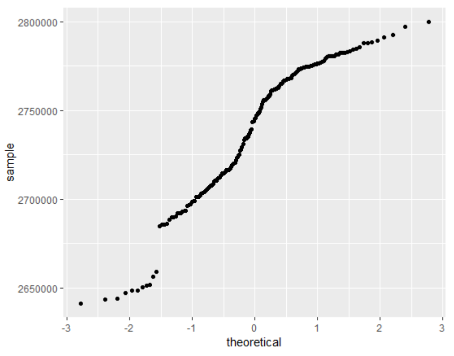
We can see that the results of Kruskal-Wallis test agree with the ones from the ANOVA test. Additionally, by conducting a pairwise Wilcox test we found matching values as the ones displayed in the pairwise t-test displayed at Table. VII, Table. VI and Table. V, which further back-up our initial assumptions and evidence.



(a) Energy consumption qq-plot



(b) CPU load qq-plot



(c) MEM load qq-plot

Fig. 6: QQ-plots on energy consumption, CPU load and memory usage

## VI. Discussion

Based on our results and the evidence to support them indicated at Section. V, we can now elaborate on them and provide guidelines based on our context.

Originally, we had specified in Section. IV that we would conduct ANOVA to test against our formulated hypotheses. However, this did not go as planned. We could not meet all the requirements specified in order to conduct a robust ANOVA test. Namely, we failed to meet the normality of our distribution assumption and the homogeneity of variance in the memory usage variable.

Fortunately, this implication did not affect the final out-

TABLE IV: ANOVA Summary

| Energy Consumption | Framework | p-value |
|---|---|---|
| | (Intercept) | < 2e-16*** |
| | Angular-ngrx-nx | 0.502291 |
| | AngularJS | 0.017488* |
| | ClojureScript | 4.52e-05*** |
| | Dojo2 | 0.000562*** |
| | React-mobx | 0.040076* |
| | React-redux | 0.199626 |
| | Svelte-sapper | 2.90e-07*** |
| | Vue | 1.78e-08*** |
| **CPU Load** | **Framework** | **p-value** |
| | (Intercept) | < 2e-16*** |
| | Angular-ngrx-nx | 0.328 |
| | AngularJS | 2.24e-05*** |
| | ClojureScript | < 2e-16*** |
| | Dojo2 | 3.79e-13*** |
| | React-mobx | 1.82e-06*** |
| | React-redux | 9.93e-06*** |
| | Svelte-sapper | < 2e-16*** |
| | Vue | 1.33e-11*** |
| **Memory Usage** | **Framework** | **p-value** |
| | (Intercept) | < 2e-16*** |
| | Angular-ngrx-nx | 1.11e-06*** |
| | AngularJS | < 2e-16*** |
| | ClojureScript | < 2e-16*** |
| | Dojo2 | 6.43e-08*** |
| | React-mobx | < 2e-16*** |
| | React-redux | < 2e-16*** |
| | Svelte-sapper | < 2e-16*** |
| | Vue | 3.13e-13*** |

come. We still proceeded with ANOVA, but did not take it as granted. After getting the results, we wanted to provide further evidence. Thus, we decided to conduct Kruskal-Wallis, a non-parametric test, that does not assume normality or homogeneity of variance. Both tests gave us the same results, leading us to build strong evidence on our case.

It is clear by now that we rejected all our initial hypotheses specified at Section. III. This means that web front-end frameworks have significant differences on energy consumption, CPU load, and memory usage.

Placing the results on the context of mobile web applications and providing guidelines for developers we can safely say that Clojurescript is the least energy efficient, with the highest CPU load and a fairly high memory usage. Hence, we can assume that Clojurescript is considered the worst choice among the examined web front-end frameworks.

Moreover, there were not any clear winners on all three aspects. If we focus on energy consumption, we can say that vue, svelte-sapper and Dojo2 are some of the most energy efficient frameworks and performance wise optimal as those frameworks tend to stress the CPU less. On the other hand, if memory usage is crucial, then angular is a clear winner followed by angular-ngrx-nx which has fairly low memory consumption between the examined frameworks.

Finally, if we had to select only one framework as the optimal choice based on energy consumption, CPU load and memory usage, and thus, provide one very specific guideline, we would choose Dojo2. Dojo2, is one of the most energy efficient frameworks, with low CPU loading and relatively low memory usage. It seems that Dojo2 combines all three aspects most efficiently from our perspective and we would suggest it for further usage.

## VII. Threats To Validity

In this section, we address threats to the validity of our experiment, following the definition given by Cook and Campbell and provide the mitigation strategies that will be used to deal with these threats.

### A. Internal Validity

**History:** History threats refer to an effect that may be observed, that is caused by an event which takes place between the pretest and the post-test and this event is not the treatment of research interest [13]. The applications are ran using the Google Chrome web browser, which uses caching to speed up recurrent future queries. The experiment will be executed multiple times, and it must be ensured that the results will not be affected by cached data. Eliminating this threat is plausible, and will be done by clearing the cache memory on the Chrome application that is installed on the phone used to conduct the experiments, before next runs of the experiment.

**Instrumentation:** Instrumentation as defined by Cook and Campbell, is a threat when an effect might be caused by a change in the measuring instrument between pretest and post-test. In the case of our experiment, to gather the necessary data required to conduct our experiment, the *Trepn* tool will be used. We operate under the assumption that the measurements that this tool will allow us to obtain, will consistently be accurate.

**Selection:** The experiment is executed only on 9 frameworks over 14. As explained in the introduction, several technical problems were found for 5 frameworks. Running the experiment only on 9 frameworks permits us to avoid large internal threats which penalize our external validity.

### B. External Validity

**Interaction of selection and treatment:** Convenience sampling which was used to choose the sample of our example also affects the external validity: all frameworks implement the same application. This represent an external threats of validity but improve the quality of our results. In this way the analysis of the results is easier due to the homogeneous sample. Analyzing data in such an easy way increases the validity of the results.

**Interaction of setting and treatment:** Our experiment was executed over a real problem. However, the scenario was defined to reproduce a possible use of "RealWorld" application. We recorded our scenario from a human interaction so it represent a possible scenario. The experiment materials are representative, indeed we ran our experiment with real frameworks used in the reality as we can see from [3]. Also the device used in the experiment represent an

TABLE V: Pairwise t-test between memory usage and web front-end frameworks

| | angular | angular-ngrx-nx | angularjs | clojurescript | dojo2 | react-mobx | react-redux | real-world |
|---|---|---|---|---|---|---|---|---|
| Angular-ngrx-nx | 1.8e-06 | - | - | - | - | - | - | - |
| AngularJS | < 2e-16 | < 2e-16 | - | - | - | - | - | - |
| ClojureScript | < 2e-16 | 3.9e-14 | 0.11095 | - | - | - | - | - |
| Dojo2 | 1.3e-07 | 0.58020 | < 2e-16 | 9.8e-13 | - | - | - | - |
| React-mobx | < 2e-16 | 8.3e-09 | 0.00018 | 0.03278 | 1.4e-07 | - | - | - |
| React-redux | < 2e-16 | 3.8e-15 | 0.22923 | 0.68941 | 1.2e-13 | 0.01157 | - | - |
| Svelte-sapper | < 2e-16 | 3.9e-13 | 0.04662 | 0.68941 | 1.0e-11 | 0.07958 | 0.45082 | - |
| Vue | 9.3e-13 | 0.00698 | 2.4e-11 | 1.4e-07 | 0.03289 | 0.00142 | 2.0e-08 | 9.1e-07 |

TABLE VI: Pairwise t-test between CPU load and web front-end frameworks

| | angular | angular-ngrx-nx | angularjs | clojurescript | dojo2 | react-mobx | react-redux | real-world |
|---|---|---|---|---|---|---|---|---|
| Angular-ngrx-nx | 0.36900 | - | - | - | - | - | - | - |
| AngularJS | 3.8e-05 | 0.00125 | - | - | - | - | - | - |
| ClojureScript | < 2e-16 | < 2e-16 | < 2e-16 | - | - | - | - | - |
| Dojo2 | 1.2e-12 | 2.3e-10 | 0.00081 | < 2e-16 | - | - | - | - |
| React-mobx | 3.6e-06 | 0.00017 | 0.59342 | < 2e-16 | 0.00494 | - | - | - |
| React-redux | 1.8e-05 | 0.00069 | 0.84572 | < 2e-16 | 0.00147 | 0.71812 | - | - |
| Svelte-sapper | < 2e-16 | < 2e-16 | 7.0e-12 | < 2e-16 | 0.00013 | 1.5e-10 | 2.0e-11 | - |
| Vue | 3.4e-11 | 5.9e-09 | 0.00530 | < 2e-16 | 0.58663 | 0.02544 | 0.00912 | 1.2e-05 |

TABLE VII: Pairwise t-test between energy consumption and web front-end frameworks

| | angular | angular-ngrx-nx | angularjs | clojurescript | dojo2 | react-mobx | react-redux | real-world |
|---|---|---|---|---|---|---|---|---|
| Angular-ngrx-nx | 0.54795 | - | - | - | - | - | - | - |
| AngularJS | 0.02880 | 0.11897 | - | - | - | - | - | - |
| ClojureScript | 0.00013 | 9.6e-06 | 4.8e-09 | - | - | - | - | - |
| Dojo2 | 0.00120 | 0.00901 | 0.31088 | 1.2e-11 | - | - | - | - |
| React-mobx | 0.06011 | 0.21137 | 0.74115 | 2.2e-08 | 0.19964 | - | - | - |
| React-redux | 0.24781 | 0.57090 | 0.31088 | 7.9e-07 | 0.04254 | 0.49028 | - | - |
| Svelte-sapper | 1.3e-06 | 2.0e-05 | 0.00702 | 2.8e-16 | 0.10009 | 0.00257 | 0.00020 | - |
| Vue | 1.1e-07 | 1.9e-06 | 0.00120 | < 2e-16 | 0.02880 | 0.00041 | 2.2e-05 | 0.58529 |

TABLE VIII: Kruskal-Wallis

| | chi-squared | p-value |
|---|---|---|
| Energy Consumption | 90.564 | 3.572e-16 |
| CPU Load | 145.13 | < 2.2e-16 |
| MEM Usage | 115.55 | < 2.2e-16 |

available device in the market.

**Interaction of history and treatment:** For our experiment internet connection represent a threat to validity. The back-end of all the frameworks is hosted online. A different kind of connection or different speed of connection could produce different results. A crowded connection could also be the cause of different results. We ran our experiment always with the same internet connection without any other traffic on the net.

## C. Construct Validity

**Construct Definition:**
As a counter-measure against an ill-defined construct, we use the GQM approach to formulate our hypotheses, as well as the dependent and independent variables.

**Experimenter expectancies:** We conduct this experiment without any intrinsic expectations or biases towards the outcome. We neither assume that there will, nor that there will not be different outcomes between the 9 different frameworks.

**Mono-Operation sBias:** This is a threat to the experiment because we attempt to keep the experiment as scoped as possible. A mitigation strategy that is deployed, is performing careful checks and restrictions in the execution environment, so to limit the possibility of external factors playing a role in the experiment. Additionally, experiments are under the threat of mono-operation bias when a single version of an independent variable is used. In this experiment, 9 different treatments are applied to the independent variable - web front-end frameworks, the different frameworks used to create the "RealWorld" application, which are created by different development teams.

## D. Conclusion Validity

**Random Irrelevancies in the Experimental Setting:** Our experimental setting is inherently prone to this threat, due to the amount of random occurrences of events that can influence our experimental procedure. Inner Android OS scheduling or background tasks running on the device, are examples of events that can tamper with the experiment results. Partial mitigation of this threat is realized in our case, by the use of a "clean" Android device, which does not have any applications installed other than those bundled from Huwaei and the Android OS. Additionally, as a proactive measure we check for, and we close any background applications that may be running, before conducting the experiments.

**Low Statistical Power:** When sample sizes are small, and the confidence level is set low, the likelihood of making an incorrect no-difference conclusion (Type II error), increases [13]. To mitigate this, we run each trial 20 times for each treatment. Moreover, we use a confidence level of 95 percent.

**The Reliability of Treatment implementation:** The experiment will be applied by four different people (authors of the paper) with different backgrounds in software engineering, mathematics and statistics, which exposes the experimental procedure to the threat of treatment implementation. The way a treatment is implemented may differ from person to person [13]. We mitigate this threat by conducting the experiment using an automated set of steps in the scenario that was described in the previous section, and using Android Runner, we will emulate human behaviour and ensure that the experiment execution is standard across all occasions of implementation.

**Random disturbances:** The conduction of the experiment will be in a controlled environment (same room) where the chance of random disturbances will be kept minimal. During the experimentation it is made sure that no other users are using the internet connection and in particular the mobile phone where the experiment is taking place. In this way we exclude other factors that could potentially introduce noise in the measurements.

## VIII. Conclusions

An experiment was conducted to determine possible differences in energy consumption between 9 web front-end frameworks, with the purpose of providing developers with guidance and information that will allow them to make an informed decision regarding this matter. Using ANOVA, we found that there are significant differences on all three variables (energy consumption, CPU load, and memory usage), rejecting all three null hypotheses included in RQ1, RQ2, and RQ3 in Section. III.

The summary of ANOVA, which we backed up by conducting a Kruskal-Wallis non-parametric test and a pairwise t-test for all subjects, revealed that different frameworks do better in certain aspects than others. However, the results show that Clojurescript is the least energy efficient, while having the highest CPU load and a rather high memory usage, rendering it a poor choice in the context of web front-end frameworks. Lastly, to provide guidance to developers, we suggest Dojo2 as an option, since it satisfies the aspect of energy efficiency, while having low CPU loading and a relatively low memory usage, achieving a balance between the three, that makes us consider it as the best option.

Regarding further extensions of the experiment, there is a potential to include a larger number of frameworks, since the RealWorld application is built using only 14, and more implementations using other frameworks are being developed. Moreover, more complex scenarios could be used to take measurements from, such as disorderly touches that could bring the application into a strenuous state, and others that include more composite app interactions, such as database calls.

## References

[1] B. Hayes, "Cloud Computing," 2008.

[2] T. Mankovskin, "Choosing a frontend framework in 2017," 2017.

[3] R. Benitte, S. Greif, and M. Rambeau, "The state of Javascript 2017," 2017.stateofjs.com/2017, 2017.

[4] M. H. Ì. C. Wohlin, P. Runeson, "Performance-Based Guidelines for Energy Efficient Mobile Applications," 2017.

[5] E. Simons and A. Pai, "RealWorld example apps," 2016.

[6] J. Schae, "A Real-World Comparison of Front-End Frameworks with Benchmarks," 2017.

[7] S. Maurya, "Top 10 Apps and Websites Developed with AngularJS," www.mobiloitte.com/blog/top-10-apps-websites-developed-angularjs/, 2016.

[8] I. Engineering, "React Native at Instagram," https://instagram-engineering.com/react-native-at-instagram-dd828a9a90c7, 2017.

[9] V. R. Basili, "Software Modelling and Measurement : The Goal/Question/Metric Paradigm," 1992.

[10] C. Wohlin, P. Runeson, M. Host, M. Ohlsson, B. Regnell, and A. Wesslen, "Experimentation in Software Engineering - An Introduction," 2012.

[11] C. Robson, "A Resource for Social Scientists and Practitioners-Researchers," 2002.

[12] S. Solutions, "ANOVA [WWW Document]," http://www.statisticssolutions.com/manova-analysis-anova/, 2013.

[13] T. D. Cook, "Campbell; dt 1979. quasi-experimentation: design and analysis issues for field settings," *Boston Houghtori Mitfliri*, 81.