Master's Thesis

Development and user experience evaluation of a tablet optimized E-Commerce app

Peter Riegler

p.riegler@student.tugraz.at

Institute for Software Technology (IST)
Graz University of Technology
Inffeldgasse 16B/II,
8010 Graz, Austria

Assessor and supervisor: Univ.-Prof. Dipl.-Ing. Dr. techn. Wolfgang Slany

Graz, August 2013

Abstract

App development today can be a challenging task. Consumers demand a solution that is functional and easy to use. Beside a top-notch design they also want to have an engaging and fun experience while using the product. And all these aspects combined define the user experience that goes beond traditional usability considerations.

In this thesis we have investigated the still very young field of mobile app development, with an additional focus on user experience on tablet devices. User experience is a critical aspect especially for consumer oriented software. Therefore we developed and evaluated a HTML5 app specially designed for the usage on a tablet computer. The application should provide an adopted interface in addition to the website. To find out how the app compares to the website in terms of user experience we conducted a summative study. A collection of state of the art practices and guidelines for mobile app development combined with experiences we collected during the development provides a great compendium for upcoming projects but also for further improvements of existing apps.

Keywords: App, Development, Usability, User Experience, Tablet, HTML5, iOS, Android

Kurzfassung

Die Entwicklung von mobilen Anwendungen (Apps) ist mittlerweile zu einer sehr komplexen und herausfordernden Aufgabe geworden. Einerseits sind die Möglichkeiten die mobile Betriebssysteme bereitstellen rasant angewachsen, andererseits sind die Anforderungen der Benutzer stetig höher geworden. Pure Funktionalität und Zweckerfüllung reichen längst nicht mehr aus um gegen unzählige konkurrierende Lösungen am Markt bestehen zu können. Neben einem gut durchdachten und herausstechenden Oberflächen-Design wird eine ausgezeichnete User Experience (UX) als das Maß aller Dinge gesehen. UX beschreibt neben der Benutzbarkeit (Usability) vor allem auch den subjektiven Eindruck des Benutzers gegenüber der Anwendung. Das Benützen der App soll Spaß machen, motivieren und leicht von der Hand gehen. Alle diese Aspekte vereint gehen weit über die traditionellen Aspekte der Usability hinaus.

In dieser Arbeit haben wir im speziellen die Entwicklung von Tablet Apps mit Fokus auf UX auf den verschiedenen Plattformen untersucht. Besonders auf Konsumenten ausgerichtete Apps wie beispielsweise bei e-Commerce Lösungen ist die User Experience ein sehr wichtiger Aspekt. Im Zuge dessen haben wir eine Anwendung mit modernen Webtechnologien (HTML5) entwickelt. Die speziell auf diesen Formfaktor zugeschnittene App wurde dann gemeinsam mit der bestehenden Webseite evaluiert. Hierzu wurden zwei geeignete Evaluierungsmethoden ausgewählt um eine Studie mit ausgewählten Probanden abzuhandeln. Aufbauend darauf konnte eine Reihe von Praktiken und Grundlagen gesammelt werden die zusammen mit jenen aus der vorhergegangenen Recherche eine Basis für weitere Projekte schaffen wird.

Schlüsselwörter: Benutzbarkeit, Performanz, HTML5, Tablet, Web, iPhone, Android, Anwendungserlebnis

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Acknowledgements

First of all I want to thank Martin Bachler, the CTO of NETCONOMY for offering me the chance to write this thesis. Joining the team and developing state-of-the-art web-applications was a great experience. I want to thank all colleagues, especially the smart-devices team for creating a friendly and welcoming working atmosphere and specially Stefan Mayer for his great guidance during the development phase. Great supporters for the written part of the thesis have been Johann Blauensteiner as well as Olivia Bronson, who both provided countless corrections and improvements. I also would like to express my sincere gratitude to my advisor, professor Wolfgang Slany for his time, support, and efforts during this thesis.

Reaching a master‘s degree is a long an challenging journey with many ups and downs. On that note I want to thank all friends and family members for their persistent support and patience during this time. Special thanks are going to Harald Tranninger who was in many ways a motivational character but also an inspiring personality and best friend. Last but not least I want to thank Bernd Bergler who shared the fate of writing a thesis beside a very challenging fulltime job. So thank you for sharing my vision and backing me up during the last months.

Graz, 15.05.2013 Peter Riegler

Contents

List of Tables

List of Figures

1. Motivation

The proliferation of smartphone and tablet devices increased dramatically in recent years, and is projected to continue growing [9][10]. Worldwide are about 1 billion web-enabled devices in use. To have all possibilities the web offers right in your pocket opened up a new world of mobile solutions and businesses[11, 20]. Current statistics of Gartner and IDC suggests that tablet device will outsell mobile PCs in between 2015 [12] and 2017 [13].

To keep up with this trend, many companies are optimizing their services for small screen devices. This includes not only the support of variable screen sizes, but also touch based navigation and the usage of available device features like GPS. The development of applications (apps) that are designed for tablets differs in many ways from conventional desktop software development and also different to smartphone development. Specific user interface and navigation paradigms should be kept in mind to ensure an engaging, intuitive and rich user experience. New sensors and hardware capabilities enable new interaction methods and convenient features. The quality of today‘s apps is much higher in comparison to just a few years ago. Present-day consumers, as a result, have high expectations. An app must be functional, intuitive, consistent, and visually appealing. The mobile development market has also caught on to these inclinations, and is adapting very quickly. This creates new challenges for developers: Staying up to speed on improvements, new trends, and the current market situation, which can be even more demanding if multiple platforms are served.

These are defining factors in the importance of the role that the web plays. It is the common denominator in all the different platforms that have emerged. Even if it is in some way the "lowest" common denominator, it is still is a reasonable possibility to develop cross platform apps via a common code base. Of course this strategy also has downsides. Native developed apps usually see better performance and can utilize all available platform-specific features. But, alternatives like HTML5 are showing promise: Many large companies are actively promoting its use. HTML5 and related mobile web technologies are designed to provide a common platform for developers to rely upon. At the same time, more and more advanced features are made available through the ongoing implementation of new HTML5 standards. In order to see what is actually possible, and how mobile web technologies relate to other technologies in this field, Chapter 4 will give insights in platforms, browsers and development strategies.

The second part of the thesis exemplifies how mobile web technologies (more precisely the Sencha Touch framework in combination with PhoneGap) were used to implement an application designed and optimized for tablet devices. The purpose of this application is an e-Commerce solution. Some methods to determine the usability and user experience of software are presented. To determine the practical outcome of our recent development, we applied some of these methods to examine our app. Finally a summative user experience study will shows how this app competes with the conventional implementation used on tablet devices.

2. Introduction

The number of different mobile platforms is enormous. Each of them offers its advantages but also disadvantages. Some platforms try very different approaches to attract customers[14]. Online marketplaces for apps offer tons of additional software. Also hardware manufacturer are developing devices with steadily increasing processing power and more and more features.

The trend of native applications, that run on a single target platform strongly dominates the big platforms Android and iOS. Their app stores contain together 99% of all available native apps.

These two market leaders are of course the primary targets for app developers. Simply because those platforms offer the most potential customers.

But for certain kinds of applications it is not the singe best way to provide a native app. Not only because it is very expensive to provide native apps for all *relevant* platforms. Also because of a problem that Tim Berners Lee explained[1] as follows. The mobile web is becoming more and more closed. This means that mobile traffic is more and more being produced by native apps. Therefore only people in these closed environments (like native apps for iOS/Android) are able to use. This produces isolated islands apart from the rest of the web. But a missing URI on top of the screen also has important consequences on social networking possibilities. Screens inside an app can not be bookmarked, shared, liked, tweeted or anything. The ability to enable all these actions on apps content pages is a major advantage of mobile web apps.

This thesis provides a collection of principles and best practices to summarize how to develop state-of-the-art web apps that offer a great experience to the user. The focus here will be the development for tablet devices. Also the differences between native apps and web apps are going to be explained. More advanced topics like usability and user experience evaluation of smartphone and tablet apps, will be the focus of the latter part of this thesis. A demonstration of a summative study will show the usability and user experience differences between a tablet optimized web-app and its desktop web app counterpart.

2.1. Research Questions

- What challenges are the developers of tablet apps facing, in particular for web-based apps?
- How can guidelines of multiple platforms be combined in a way to design for the best possible user experience across them?

[1] `http://www.zdnet.com/apps-no-root-your-device-serves-others-berners-lee-7000010661/`

- What must be considered to get an engaging and stunning user experience on a tablet device?
- How does the usability and user experience of a tablet optimized e-Commerce app compare to the desktop version?

2.2. Motivation

- Reflect current known practices and guidelines (native ones as well as guidelines for web apps) and define them in the context of our application.
- Review usability and user experience evaluation techniques and their applicability for the context of small screen user experience testing?
- Analyse differences of user experience and usability measures for smartphone and tablet app examination. Comparison of the developed tablet optimized app, to the web app mainly focused on desktop usage?

2.3. Outline and Scope of this Work

This work is aimed at improving user-experience as well as user experience evaluation of tablet apps. The first chapters will give an introduction about today‘s landscape of mobile platforms and their eco systems. This includes an introduction to the most important operating systems, mobile browsers and app stores. Also the web can be seen as platform and therefore in Chapter 5 the upcoming HTML5 standard will be examined in more detail.

An preface to the development of mobile apps (Chapter 6) is followed by usability concerns (Chapter 7). We also developed an e-Commerce app using web technologies and some insights into the development process is given in Chapter 9. Beside the development with web technologies also native development for Android and iOS will be briefly introduced. Some of the presented usability guidelines and design principles are similar or even identical on those platforms. Others are completely different and that is where web apps typically are facing their biggest challenges. Chapter 8 covers methods for usability and user experience evaluation to provide a profound background for the practical part (Chapter 9) of this work. It contains a documentation of the development process and the modus operandi for the usability/user experience study. The study demonstrates a side by side comparison of the tablet optimized HTML5 app and the conventional desktop web shop. The results and insights we could make due to that have been very valuable for our ongoing but also upcoming app development projects. By using an additional questionnaire, user experience measures have been collected to provide more holistic insights. So the main goal for the study is a summative one.

This thesis will not provide any new measures or questionnaires. It demonstrates how we implemented an application and verified certain aspects of the user experience by applying well known and proven concepts. Further economic considerations like Return on Investment (ROI) will not be covered either.

3. Related Work

Some existing works provided some direction for this thesis. But as Nayebi et al. [15] wrote recently "*There is no scientific research addressing the requirements of new mobile user interfaces*". This shows a major issue when doing research in this field. Although the internet offers a vast number of sources around this topic, real scientific sources are rare. Nevertheless I will provide some related work that has been found valuable in this section.

The "Journal of Industry, Competition and Trade" contains the article of Martin Kenney et al. [3]. They analysed the eco systems and market strategies of major mobile device vendors. They also determined the evolvement of mobile internet competitions and gave an outlook over the possible future development in this field. The focus of this article has been the strategies of the four big players Apple, Google, Microsoft and Nokia. Nowadays app markets are an important part of those mobile eco systems.

In his master thesis [16] Kimmo Puputti highlighted the problems todays mobile development faces. Platform segmentation on the mobile market makes it hard to build applications that run on a broad range of devices. Therefore he examines the most promising future way of cross-platform mobile app development. In the eyes of Puputti this will be the web as common platform. With HTML5 the web will get more powerful, more user-friendly and it will offer more possibilities for mobile developers. He also mentioned new tools and frameworks that bring additional benefits to developers and users of mobile web applications.

A brief report by Rauche et al. [17] addressed user experience evaluation of mobile apps. Rauche also did a summative study with the AttrakDiff questionnaire. The selection of

Usability Testing of Mobile Applications by Kaikonnen et al. [18] compared the method of field testing against laboratory testing. Regarding their outcome, field testing for mobile usability test is only worth doing for studies where user behaviour in an unbiased environment is the center of interest. Although the results have been published in the year of 2005, the main messages are still valid to a great extent until today.

What kind of usability inspection methods have appeared since the early days? Jakob Nielsen summarized his findings about this topic in a journal article [19] back in the year of 1994. Many of these methods are still in use today, often in updated formats, but the core ideas behind them remain a foundation.

Coursaris et al. [20] introduced an adapted contextual usability framework. They adapted their framework proposed in [21] to fit special needs of for a mobile environment.

Very closely related to the first part of thesis is a work with the title "User Centered Cross-Platform Application Development for Mobile Devices" by Jana Mrazova [22]. She compared native to cross-platform applications and studied user behaviour to create usability guidelines

for cross-platform mobile app development. Although the technical insights are not very deep the general subjections are leading developers and designers to the right direction.

An article written by Brennan Browne [23], summarizes experiences he made during a year of tablet UX research. Browne is a design/user experience researcher at Facebook. His focus has been the iPad and how people are using it. The usability and the context of use differences between tablets and smartphones. Tablet usage falls along the lines of a small, convenient computer. It is often still a shared device and not as person as for example a mobile phone would be. Furthermore Browne writes that users do not bring their tablets everywhere, like they would a phone. Due to the screen size, users expect a full-featured app or website that offers more possibilities than its smart phone counterpart. They also concluded that tablet users are more concerned about security and data integrity.

Another similar topic covers the thesis of Sami Pekkala [24], who conducted a "Usability evaluation of design solutions for tablet magazines". The formative evaluations as well as a summative evaluation brought insight in usability measures for popular magazine apps. At the closing, he summarized suggestions and design solutions for applications of this type.

Zmags a rich-media marketing platform vendor gave out a white paper in February 2012 [25]. They formed an analysis on the performance of the top 100 Internet retailers in m-Commerce (in addition to e-Commerce on mobile devices). Surprisingly, this revealed that only a few of them are offering solutions that make full use of today‘s m-Commerce potential. This is very alarming when viewed alongside growth-rates in this field.

Infosys, a global leading consulting company, published some short but very interesting articles in their white paper "Advances in User Experience Design" [26]. They highlighted the importance of user experience but also mentioned possible security concerns of HTML5. One of the contained articles also focuses on the connection between mobile commerce and the user experience.

Naumann et al. [27] also conducted a comparative usability study. In contrast to this work, they compared three different categories of devices (PDA, tablet and PC). They used the SUS and the AttrakDiff questionnaire, beside the SUMI and SASSI questionnaires. Their evaluation showed that the AttrakDiff provides good results for Multi-Modal interfaces.

4. Mobile Eco Systems

When a consumer decides to buy a specific smartphone or tablet, the device isn't the only thing that must be considered. Each platform vendor also offers a mobile eco system for their users and developers. This includes the mobile platform itself, consisting of the hardware (smartphone or tablet device) and the Operating System (OS).). In addition to this, developers are able to write separate applications to run on these devices. Therefore vendors are offering SDKs and tools for development. They provide app stores for distribution and monetization. Additional services like cloud file storage and data synchronization (calendar, email, etc.) are provided. All these pieces defines a particular mobile eco system. This not only brings powerful features and possibilities, but also ties more individuals to certain companies. For developers, the mobile trend opened up a whole new market.

So which platforms are important for developers in terms of market share, compatibility or future trends? This question will be answered in the following section, by introducing today's most important mobile platform vendors and their eco systems. In order to show our insight of the current market situation, we are also going to provide some numbers.

4.1. Mobile Operating Systems

When we look at the current market situation in terms of shipped devices by hardware manufacturer, Apple and Samsung are the predominant companies on the global market. But for the software developers the distribution of the Operating System (OS) is much more important. We will briefly introduce them in Section 4.1 to provide an overview and point out their relevance on the global market.

operating systems is dominated by two major companies, namely Google and Apple. Each of them offers a version for smart phones as well as tablet devices. Their operating systems, Android and iOS, are currently accounting for a total market share of 90% [1] in the smartphone market and 94% [2] on the Tablet market. That is why it is usually sufficient for many companies to focus their mobile strategy only on these two vendors. The dominance of iOS and Android was also the main reason behind focusing the content of this thesis on those influencing platforms.

Smartphone Market Situation
The following numbers are reported by the International Data Corporation (IDC) and reflects data provided by the IDC Worldwide Mobile Phone Tracker [1]. Thus only smartphones and no tablet devices are contained here. This data reveals the numbers of sold devices for different types of operating systems in the 3rd quarter of 2012. Apple iPhones with the proprietary iOS reached a market share of around 15%. The undisputed market leader at the time of writing is Google's open source OS Android. In Q3 2012 Android reached a market share of

more than 75% in terms of sold smartphone devices [28]. In contrast to iOS, it powers the smartphones of several different hardware manufacturers, including Samsung, HTC, Asus, LG, among others. These companies are part of the Open Handset Alliance (OHA), founded by Google and 33 additional companies on November 5th, 2007 [29]. The goal of the OHA was the development of innovative and improved mobile devices by following an open, collaborative approach. These companies united their forces to compete with Apple, the market leader at that time. The following operating systems also reached a notable market share: Symbian the OS by the former market leader Nokia fell down to 2.3%; BlackBerry OS again with heavy losses could only reach 4.3%; and the desktop giant Microsoft reached a market share of 2.0% with Windows Phone.

About two years ago Symbian and RIM (recently renamed to Blackberry) played an important role on the smartphone market, but they continuously lost market shares. Windows Phone the successor of Microsoft's Windows Mobile on the other hand struggle with catching up to Android and iOS. They launched in 2010 so 3 years after the first iPhone was presented and till now they were not able to gain a 2 digit market share. But analysts predict a possible future growth, due to tight integration of the desktop ecosystem. Ramon Llamas, research manager at IDS said that Android will probably stay in front, although competitors gaining market share will mainly target the current market leader.

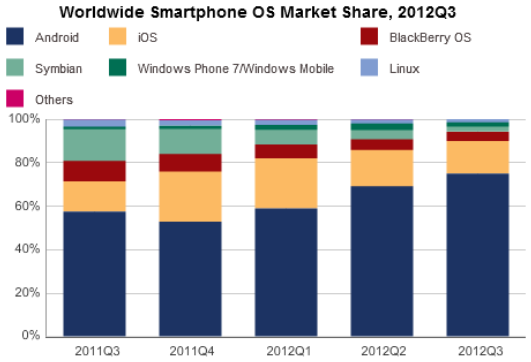

Figure 4.1.: IDC Worldwide Smartphone-OS Sales, Q3 2012 [1]

Operating System	**Market Share 2012Q3** [%]	**Market Share 2011Q3** [%]
Android	75.0	57.5
iOS	14.9	13.8
BlackBerry	4.3	9.5
Symbian	2.3	14.6
Windows Phone	2.0	1.2
Linux	1.5	3.3
Others	0.0	0.1

Table 4.1.: Smartphone OS Sales, Q3 2012 [1]

Tablet Market Situation

On the tablet market Apple is still market leader. In October 2012 Neil Mawston from the "Strategy Analytics Tablet & Touchscreen" (TTS) reported that Android reaches 41.3% of the overall tablet market share and Apple dominates with around 57% [2] respectively 61.5% [30].

In comparison to the numbers a year ago, this is indicative that Android is finally starting to compete with iOS on the tablet market. With growth in sales 49.5% on the worldwide tablet market between Q3 2011 and Q3 2012 [2], this showed how important this market will be in the future. Since the release of Windows 8, it could not gain a high market share, but it is already shipped with many new devices. Microsoft's own device the Microsoft Surface will be shipped with their new touch optimized OS flagship [31].

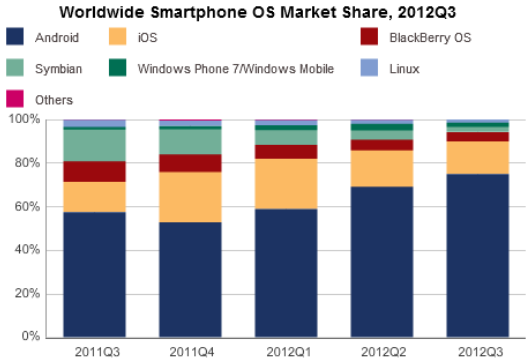

Figure 4.2.: IDC Worldwide Tablet-OS Sales, Q3 2012 [7]

Operating System	**Market Share 2012Q3** [%]	**Market Share 2011Q3** [%]
Android	41.3	29.2
iOS	56.7	64.5
Microsoft	1.6	2.3
Others	4.1	0.4

Table 4.2.: Tablet OS Market Share, Q3 2012 [2]

Because the previous stated numbers only reflect the last quarter, we also want to present the market share data by `http://www.netmarketshare.com`[32]. These numbers contain worldwide tablet as well as smartphone sales, representing the years 2011 – 2012. In this time period Java ME, powering mid to low-end devices, was selling pretty well but these devices mostly represent feature phones.

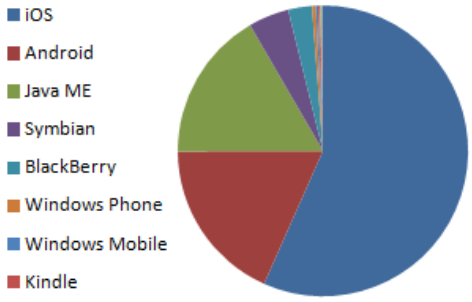

Figure 4.3.: Total Market-Share of Smartphone and Tablet OS (Years 2011 – 2012)

Operating System	**Market Share** [%]
iOS	56.6
Android	18.3
Java ME	16.6
Symbian	4.5
BlackBerry	2.7
Windows Phone and Windows Mobile	0.7
Others	0.54

Table 4.3.: Total Market-Share of Smartphone and Tablet OS (Years 2011-2012)

Facing domination on the tablet market, it is obvious that many mobile business strategies are mainly focusing on iOS and Android. Both platforms offer powerful development tools and SDKs. But, in contrast to desktop operating systems, they are very different. A native app for those two platforms has to be developed separately, without a common code base. It is obvious that is a very expansive. That is why cross-platform development frameworks occurred. Details on that subject will be given in Chapter 6. In my seminar thesis we covered some of them in detail. Products like Appcelerators Titanium, Rhodes or Xamarin came up to build apps across multiple mobile platforms. Nevertheless as Julian Ohrt et.al. wrote, they all have still a long way to go to compete with pure native apps [33]. So the most promising solutions in the long run will be the usage of HTML5 web technologies. What exactly is possible with HTML5 at the moment and what might it bring to the market in the near future? To get answers on some of these questions, refer to Chapter 5. Hybrid solutions can also be used to address the lack of device feature support and native distribution. Consult

Section 6.1 for more information about this class of apps [34][35].

4.1.1. The Web as Application Platform

Something all platforms have in common (not only mobile varieties), is the ability to render web content in a browser window. That multi-platform capability is the biggest strength of the web. And now, with the new HTML5 standard, new features for mobile apps are planned, and several of these are already implemented in today‘s mobile browsers. The standard is not completed yet, but the support of hardware features like cameras, offline storage, geolocation, and multi-touch has already been implemented on many platforms [36] [37].

What advantages come with the development of apps that use web technologies? Most importantly, only a single app for all platforms is necessary. That means most of the knowledge, processes, and tools will be shared. Some additional changes and platform specific optimizations on portions of code might be necessary. But, nevertheless, much of the development costs can be spared if only a single codebase has to be developed and maintained. Saving costs and time are especially important for companies that already have experience in web development. This also means a faster ship date to market is possible.

Drawbacks, in contrast to native apps, are differences in the appearance of the user interface. This is because most web apps are not meant to imitate native UI elements. To do so, a platform specific design adoption would be needed. This avoids differences to platform specific UI-guidelines. A very promising framework by Telerik that fulfils this task is Kendo UI [38]. But, nevertheless, a typical web-app does not perform as well as a native one. A wiki post on the W3C site [39] tries to explain the core problems behind this issue. They identified the main bottlenecks for web-apps as:

- Missing or poor GPU acceleration
- Too much work for the main threads
- Slow DOM interaction performance
- Poor browser performance

4.1.2. Upcoming Web-Centric Operating Systems

Some Linux based mobile operating systems have appeared, and many of them offer the ability to run HTML5 apps outside a browser just like native apps. This concept has been utilized extensively by Firefox OS, Tizen, webOS and Ubuntu Touch. Firefox OS fully relies on the HTML5 stack while others support development of native apps as well. To show differences to other platforms, we will provide more insights into Mozillas approach. The late start will make it difficult to gain a notable market share, but the approach to base all apps on HTML5 web technologies is still a very promising one. Also "Wireless Smartphone Strategies" (WSS) service forecasted Firefox OS to gain 1% of market share during the year 2013 [40].

Firefox OS:
As an alternative to conventional platforms Mozilla released its own mobile operating system (OS). Mozilla attempted to create a very transparent and open platform, built upon web

standards. The open source Linux based OS can run on a standard Android phone. Firefox OS compatible phone are sold on `http://geeksphone.com`.

To test apps during development there is the Firefox OS Simulator that runs on a desktop computer. The simulator is a Firefox Add-On containing the simulator itself, a dashboard to manage apps, and debugging features. Mozilla also integrated some Developer tools that are pre-installed in the browser. This includes a DOM inspector, a CSS style editor as well as a tool to preview the look and feel on different screen sizes.

Two basic kinds of apps run on Firefox OS: Hosted apps and packaged apps. The first type of app is essentially a websites that relies on a connection to a web server that hosts the app. It is executed inside an app context but relies on a (permanent) internet connection. The second kind (packaged) contains all resources like images, markups and scripts packaged inside the app and can therefore be executed without an internet connection. Nevertheless these apps may update themself or their content when a connection is established.

For developers it is important to find users for their apps, and similarly, users want to easily find new apps for their devices. Mozilla‘s Firefox Marketplace serves both of these needs. Apps that are published via this market have been reviewed by Mozilla to ensure quality and prevent malicious apps. Developers should be aware of these review criteria [1] and UI guidelines [2] are available online.

The possibility to reuse existing knowledge of web development is of great benefit for (web) developers, and Mozilla is still one of the dominant players in the world of the open web. For endusers with an outdated Android device, Firefox OS might offer a suitable alternative– the broken Android update strategy offers possibilities for competing platforms to gain market shares.

4.2. App Stores

An integral part of each mobile platform, other than the OS itself, is its respective app store. After a basic introduction into the concept itself, we will highlight the main differences between the competitors. The focus will rely on aspects that are relevant for developers.

An app store is the place where users can discover and download apps for their device. A central way of distributing software to end users, is known in the Linux world under the term package management system for almost two decades. This model has been adopted and extended with social and commercial aspects. This opened up a new and very successful business model. For both sides, developers and of course also for platform vendors[3].

But what exactly is the business model behind this? The answer is far more complex than just selling apps for a fixed price. Additionally, monetization in app purchases offers the possibility to turn profit. But today, apps are not the only commodity that is distributed via these stores. All types of content, such as magazines, music, movies, etc. can be purchased here. This

[1] Mozilla mobile web-app review criteria `https://developer.mozilla.org/en-US/docs/Web/Apps/Publishing/Marketplace_review_criteria`

[2] Mozilla UI guidelines for mobile web-apps `https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/UX?redirectlocale=en-US&redirectslug=Web%2FApps%2FUI-Guidelines`

brings the entire mobile ecosystem to a new level, where content and its distribution are the most important key values.

There are two opposite strategies for app stores: open and closed approaches. Most companies have chosen to elect a closed approach that is colloquially known under the term "walled garden". Apple‘s App Store, Microsoft‘s Windows Phone Store, Blackberry‘s App World, among others, allow the distribution to end customers solely via their own store. This restriction is also important to know as developer: If a vendor decides that an app does not fulfil their (content) guidelines, they can ban it from their store. This approach creates some controversy, because it creates a sort of monopoly inside the platforms ecosystem. Vendors, on the other hand, can promise a higher quality product for the end-user.

The open approach, utilized by Google, also allows competitors to provide apps and content. Apart from Google‘s Play Store, there are several other options available. The Amazon App Store is one of the more prominent alternatives, and apps can also be distributed via any other channel. The application package file (.apk file) can be distributed via email or made available for download from a website.

The next thing that sets apart today‘s app stores is the approval process for apps (also known as prescreening). This process can be very rigorous, using Apple‘s procedure as an example. On the other side of the spectrum, Google only minimally screens content, and violating apps are usually only removed if they turn out be malware. The quality of apps is measured primarily with user ratings and comments. Prescreening has a large amount of influence on the duration of the approval time. While an iOS app might take several days to weeks to be made available for download, an Android app is typically published in a few hours [41].

What other aspects of app stores are important for developers? It's of course, important to note the costs that are involved in the publication and distribution of apps. There are three fees that are relevant:

- An initial registration fee:
  This fee is a one-time payment that should cover account setup costs.
- The regular subscription fee:
  Some store providers claim an annual subscription fee.
- And the transaction fee for each sold app:
  This is typically 30% of the app price.

An overview of the most important app stores is shown in Table 4.4[14]. The relevant Ovi Store of Nokia was left out because of the company's recent decision to use Windows Phone as their primary future OS[3]. Therefore the Ovi Store will be merged into the Windows Phone Store[4]. The huge market share of Android brought out many competing app stores. But considering the current relevance only Amazons app store will be listed here. This store also provides apps and content for all Amazons Kindle Fire devices. They run a forked version of Android [42, 177] and have the Amazon App Store preinstalled. Because of the very closed marketing strategies there have been no numbers published for the total app downloads using either the Amazon App Store or the Windows Phone Store.

Vendor	**Name**	**Platform**	**Type**	**Registration-Fee**	**Subscription-Fee**	**Transaction-Fee**	**Apps Available**	**Download Count**
Google	Play Store	Android	Open	25$	-	30%	> 700.000	25 B
Amazon	App Store	Android	Open	-	-	30%	> 50.000	-
Apple	App Store	iOS	Walled Garden	-	99$	30%	> 700.000	35 B
Microsoft	Windows Phone Store	Windows Phone	Walled Garden	-	99$	30%	> 125.000	-
RIM	BlackBerrry AppWorld	BlackBerry	Walled Garden	-	-	30%	> 105.000	2 B

Table 4.4.: Overview of App Stores [3][4] and additional online sources

[3] http://www.developer.nokia.com/Develop/
[4] http://arstechnica.com/gadgets/2011/02/nokia-adopts-windows-phone-7-as-primary-platform/
[4] https://developer.apple.com/programs/ios/
[4] http://www.theverge.com/2012/9/6/3296612/amazon-appstore-for-android-50000-app-count-september-2012

News: Starting August 2013, the Amazon app store started distributing HTML5 web-apps in addition to native apps for their kindle fire devices[5].

Many app stores provide a rating mechanism for users. This provides an easy way to define their impression about the quality of the app. The combined results of a large number user ratings are usually reliable indicators on app quality. Additional comments can fill in the gaps for information about common errors that occur, lacks of features or missing device support. Nayebi et al. [15] stated that these ratings can be very useful for usability studies. Indeed, it is perplexing that they are not used in more of them.

4.3. Mobile Browsers

Mobile browsers are one of the most important tools on every smartphone and tablet today. This chapter not only gives an overview of the most important mobile browsers, but also explains some of their pros and cons in contrast to competitors. In addition to the preinstalled browsers of platform vendors, widely-used downloadable browsers will also be covered. The summary at the end of this section additionally points out their current market shares[6] and HTML5 capabilities due to `html5test.com`[43].

4.3.1. Safari

Mobile Safari is the preinstalled browser on iOS devices. It is a Webkit based browser that already contains broad support for HTML5. It had been developed to be used as easy as possible by using touch and multi-touch navigation. The iPhone/iPod Touch version is different to the Safari installed on the iPad Firtman et.al [37] writes that in September 2012 Mobile Safari has a market share of 90% on tablet computers. There exists a strong coupling between the Safari Browser and iOS. Thus Safari cannot be updated via the App Store. Updates are done together with a new version of the operating system. This decreases the flexibility for updates, which is a disadvantage in comparison to non-coupled browsers [37, 47-48]. The overall performance of Safari is very good, with a score of 386 at `html5test.com`[43], it shows an exemplary support of HTML5. More details about the currents state of HTML5 support will be covered in Section 5.4[37, 50-54].

4.3.2. Android Browser and Google Chrome

The Android Browser is like Safari based on Webkit. In terms of performance the Android Browser has always been far behind Safari. Some hardware manufacturer even shipped their devices with a customized version of the stock Android browser (Samsung, Sony, Barnes, ...). Developers hoped that the mobile browser would be similar to the Google Chrome desktop browser, but this was not the result. In order to address these issues Google introduced a

[5] `https://developer.amazon.com/post/Tx2HGWRGOW5YG8/Amazon-is-Now-Accepting-HTML5-Web-Apps-Making-it-Easy-for-Y` `html`

[6] `http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1` accessed on the 19th Dec. 2012

mobile version of Chrome in 2012. Chrome for Android is available on devices running Android 4.0+ and replaced the default browser since Android Jelly Bean (4.1). The new browser offers a better performance in terms of scrolling and rendering. More on that in Chapter 9. Chrome for Android shares most of its code with the desktop version and instant updates via the Google Play Market made it one of the most powerful and up to date HTML5 browsers. With a score of 390 on `html5test.com`[43] Google could achieve the 2^{nd} highest ranking at the time of writing. Although Chrome was launched for iOS as well, this implementation is not based on Chrome but on the default iOS Webkit engine [37, 48].

4.3.3. Internet Explorer

With Windows CE in 1996 Microsoft launched one of the first mobile browsers. Pocket Internet Explorer as it was called. Today‘s version is simply called Internet Explorer. Microsoft chose the same name for desktop and mobile versions, because they share the same rendering engine. Windows Phone 8 includes the newest Internet Explorer 10 release that brings a lot more HTML5 support than its predecessor. Nevertheless the Internet Explorer on Windows Phone 8 devices still has the lowest rate of HTML5 feature coverage among preinstalled browsers (score of 320 [43]) [37, 50-54][44].

4.3.4. BlackBerry Browser

With BlackBerry 10 RIM offers a completely redesigned and reengineered Webkit based browser that brings the power of current HTML5 features to the latest BlackBerry devices. While the underlying OS is not predicted to good enough to save the drowning company [7] [8], the included browser breaks all records in providing one of the best HTML5 capabilities ever seen. Though still in development, the score of 484 [43] at the time of writing is definitely worth noting. This is underlined by the announcement of RIMs Technical Director of Web Technologies, Matthew Staikos. He wrote[45] that their newest browser could achieve to meet the Ringmark Ring 1 benchmark standard. [37, 50-54]

4.3.5. Downloadable Browsers

Opera Mobile and Opera Mini
The desktop browser entered the mobile world and released downloadable browsers for Android, MeeGo and Symbian devices. With a score of 406 [43] for the version 12.10 of Opera Mobile the Norwegian Software company released a stable version of their browser with the highest score in this ranking. With the performing Presto rendering-engine and data synchronization to the desktop browser opera mobile is offering a great alternative to preinstalled browsers. A unique feature offered by opera is the transcoder that pre-compresses web pages before they are delivered to save bandwidth. Of course on new devices with high bandwidth data plans and unlimited traffic this becomes less useful [44].

[7] `http://www.rethink-wireless.com/2012/11/07/blackberry-10-doa-analyst.htm`
[8] `http://news.yahoo.com/blackberry-10-predicted-flop-could-result-samsung-acquiring-135047533.html`

Firefox
The downloadable Mozilla browser is available for Android, MeeGo and Windows 8. The already mentioned Firefox OS even goes one step ahead and uses the browser engine as the core of the platform itself. Like the desktop browser it uses the Gecko rendering engine. In version 16 Mozilla stepped on the 3rd place of current browsers with a score of 388. The latest version also includes a synchronization mechanism between devices. For Android devices the latest beta as well as alpha version (named Aurora) can be downloaded via the Google Play Store [43][31].

4.3.6. Summary

Table 4.5 shows an overview of mobile Browsers and their main characteristics (rendering engine, HTML5 support score and market share[9].

Browser	**Current platforms**	**Engine**	**Score**	**Market Share**
Safari	iOS	Webkit	386	61.50%
Android browser	Android up to 4.1	Webkit	297	26.09%
Chrome	Android > 4.0	Webkit	390	1.14%
BlackBerry Browser	BlackBerry OS 5.x	Custom	-	-
	BlackBerry OS 6.0, 7.0	Webkit	288	-
	Tablet OS / BB10	Webkit	No	1.09% total
Internet Explorer	Windows Phone	Trident	320	0.95%
Firefox	Android, MeeGo, Firefox OS	Gecko	388	0.01%
Opera Mobile	Android, Symbian	Presto	406	0.53%
Opera Mini	Android, Bada, iOS, Symbian, Windows Mobile	Presto	406	7.02%

Table 4.5.: Overview of Mobile Browsers [37, 65][43] and Market Shares for November 2012

Measurements of market shares for browsers are typically done by analysing the usage characteristics on many important websites. So of course the numbers vary between different statistics providers. They are depending on the actual metrics and the amount of data that was analysed. As we could see, the vast majority of mobile users are browsing the web via a Webkit based browser [37]. Considering this fact, it is probably reasonable to focus on this browser family.

[9] `http://www.netmarketshare.com/browser-market-share.aspx?qprid=0&qpcustomd=1`

4.4. Device Types

4.4.1. Smartphone

Definitions for a smartphone differ in literature and the term has been redefined for several times over the last years. There are no hard boundaries between feature phones and smartphones. What all definitions have in common is that a smartphone is offering the functionality of typical feature phones with additional ones. This means that smartphones can be seen as a combination of a cell-phone and a PDA. The term PDA became popular in the mid-90s and represented mobile devices that could be used as personal information manager. The minimum requirement today for a Smartphone include: a multi-tasking OS, a modern browser and high-speed wireless connectivity (WiFi, 3G/4G connections). Also some of the following features are commonly available:

- Large screen size
- Video-capable camera
- Advanced input technology (touchscreen, QWERT keyboard, stylus)
- Advanced computing capability
- Large amount of storage
- Sensors (gyro, compass, accelerometer)
- GPS
- 2D/3D graphic acceleration

In today‘s terms, "smartphone" often mean a phone with a touchscreen. This new input technology has greatly innovated interaction with these types of devices. Screen sizes vary from the typical 2”, up to 5” diagonally. Apart from this hardware categorization, what really makes them "smart" and useful are the appropriate software tools. Small applications (apps) of any kind enable the user to complete certain tasks. This opened up yet another new market for software developers, that are now able to sell their applications via online marketplaces (see Section 4.2). All this possibilities combined are what made this era of mobile devices so successful. Because of the relatively small screen size in contrast to desktop computers, interface design for this devices it very different. With a touchscreen as the main input element, this brings additional considerations for interaction design and usability [11, 8-12][46, 8].

4.4.2. Tablet

The success of Smartphones paved the way for a new kind of device: the tablet computer. A tablet is a thin, touch based computing device, with a typical screen size of 7” and more. Often a tablet is seen as a bigger Smartphone without the capability to make phone calls. This isn‘t entirely untrue, but observations of usage and interviews with long time users are telling a different story. Research gathered by AnswerLab [23] investigating iPad users, showed that tablets are most often used as a convenient alternative to desktop computers. This means that they are gradually replacing desktop computers on the consumer market. Tablet sales are accounting for enormous growing numbers. Gartner [10] wrote of a 47 increase between 2010 and 2011. The growth for conventional desktop computers and notebooks is on the decrease [47].

The mobility of this device type is in between Smartphones and laptop computers. They are not as mobile and personal as a phone. Most of the time users keep their tablet at a fixed location like their flat, workplace or so on. They only bring their tablet if they know they will need it, e.g. for a train ride, at the university or in a cafe. Here it serves as a portable, easier and convenient way to use a computer. So in many cases users are bringing their tablet instead of their laptop computer. And interesting aspect is that tablet users often use the web in the first place, even when equivalent native apps are available. This is very different to the typical Smartphone usage. A field study by Browne et al. [23] explains this behaviour. The iPad for example, provides great browsing behaviour (much better than on smartphones), so as a result most users‘ first step is to open up the web browser and either search for the desired keywords or open up a bookmarked website. But nevertheless, for regularly used apps that provide a user experience that goes beyond the one of the web version, users tend to install a native app. Because of the fact that tablets introduce a new kind of interaction device, a lot of new guidelines, patterns and paradigms have to be considered by developers to provide the user with a great experience [23][48, 132].

5. HTML5

HTML (Hyper Text Markup Language) specifications are made by the World World Wide Web Consortium (W3C) [49] and the Web Hypertext Application Technology Working Group (WHATWG) [50]. Both organizations provide information about the current standard on their websites. It is important to note that HTML5 is not yet a finished standard. Also the newly introduced "Living Standard" development model implies that it will always be evolving. This means that single definitions reach specific states: *Idea*, *First draft*, *Working draft*, *Last call for comments*, *Awaiting implementation feedback* followed by *Implemented* and finally *Widely deployed*. There are official snapshots of this the specification available, as well as drafts of currently non-stable definitions. The W3C and WHATWG websites also contain information about the current implementation status among major browsers.

The HTML5 standard has become a big buzz word in recent years. The successor of HTML 4.01 is planned to bring web development to the next level. It introduces new markup elements and attributes and very powerful Application Programming Interfaces (APIs) to build next generation web apps. This will improve the experience for users on the web and brings also more possibilities for semantic structuring. Features that were only applicable for desktop software are now possible on the web. Design principles for the development include utility, interoperability and universal access [51]. This is why HTML5 could succeed XHTML 2.0 and other mobile markup language variants [44]. Handling faulty code as effectively as possible is additionally one of the key success factors of this standard [36]. Error tolerance beats strict validation. All major browser vendors are actively supporting HTML5 and continually implement new HTML5 features. With major browsers, the W3C means Safari, Chrome, Firefox, Opera and Internet Explorer. The current releases of browsers already support a broad spectrum of stable defined specifications, and even impermanent ones, like the offline storage feature [36][37][46, 14].

The next two chapters will give more insight about new markup elements and features introduced by this standard. But how do we know what we actually can use for mobile web projects right now? Therefore a summary about how far the implementation of this features in current mobile browsers are is following in the last section of this chapter.

5.1. New Markup Elements and Attributes

Until now, 28 new markup elements have been introduced with HTML5. Some of them aim to provide a more semantic method of structuring content. This is important not only for search engines, but for site navigation. The browser gets more knowledge about the structure and semantic and for example becomes the ability to jump to specific parts like the side navigation.

Other introduced elements offer possibilities to include multimedia data. We are going to divulge some of them very briefly in this section.

Extensions for form elements (`date, time, url, email, color` etc.) and connection of elements to forms via the form attribute. Due to this types client side validations are possible. Furthermore, mobile devices can change the input method (software keyboard) according to this type [36]. New attributes improve form elements with new possibilities: `autocomplete`, `min`, `max`, `multiple`, `pattern` und `step` [37][36].

Audio- and **Video-Elements** introduce the usage of audio and video data that does not rely on the installation of additional plug-ins like flash [36].

The **Canvas-Element** offers a very powerful way of scriptable 2D drawing. This huge specification got its own separate document but still belongs to the other HTML5 definitions. It can be used for interactive backgrounds, diagrams and even games. These kinds of applications have previously been possible through technologies like flash, but this will allow availability on (mobile) devices even if they do not support flash [36].

5.2. New JavaScript APIs and related standards

Despite the new markup offered by HTML5, there are also new JavaScript APIs available. Markup alone cannot create all of the magic happening inside web apps and websites; dynamic and interactive implementations are done in JavaScript. The new APIs enable a richer and more interactive experience for the user, and some of them are specially designed to improve the user experience on mobile touch-based devices. We will explain some of them very briefly in this section.

5.2.1. Web Storage API

By now defined in its own specification[1] the Web Storage API can be seen as an capacity improved version of cookies. It offers the ability to store up to 5 MB of key-value data and is separated in session storage (available in a single window until it is closed) and local storage (bound to a domain, stored permanently)[36].

5.2.2. Web SQL Database API and Indexed Database API

A SQL powered database that can be accessed via a JavaScript API is the basic idea behind Web SQL Databases. The maximum size of the database has to be chosen during its initialization [36]. Because it does not have any other implementations than SQLite it has not been accepted as a standard by the W3C[2]. Nevertheless it has been widely adopted in mobile environments and is available in Safari on iOS, Chrome for Android, the Android browser, the BlackBerry browser and Opera Mobile.

[1] `http://www.w3.org/TR/webstorage/`
[2] `http://www.w3.org/TR/webdatabase/`

Alternatively, some browsers have also chosen an Indexed DB implementation, a NoSQL database API which is also currently a W3C *Candidate Recommendation*[3]. Even though Web SQL is currently available on all major mobile Browsers it will eventually be replaced by Indexed DB [31].

5.2.3. Drag and Drop

DDrag and Drop capabilities are implemented in many browsers via an API originally introduced by Microsoft. Although the API is said to be the black sheep among the others, it made its way into the HTML5 specification[4]. The ability to drag and drop content inside the browser, but also between the browser and other applications is not only a powerful tool but also very intuitive for users [36].

5.2.4. Offline Support

An important feature for mobile devices is offline support for web apps. The W3C refers to two solutions for that matter: A local database, as described in Section 5.2.2 or an offline application HTTP cache[5]. This cache implementation will provide a local copy of resources that is defined in the applications manifest file. The local cache is accessible via the `applicationCache` object [36]. Due to its relevance in the mobile context, the availability along mobile browsers is very high. The complexity involved in implementing an application cache for a mobile web app is very challenging for developers [31].

5.2.5. HTML5 Web Messaging

HTML5 web messaging allows the exchange of messages across domains. The exchange of data between domains is currently prevented by browsers to avoid cross-site-scripting (XSS) attacks. In order to allow the exchange of data, the right security measures should be placed beforehand [36]. The CORS (Cross Origin Resource Sharing) specification outlines how to enable a script to access different domains. Web Messaging should imply a new possibility to exchange information across documents without enabling XSS attacks [31]. The W3C site[6] contains the current state of this feature which is a W3C Candidate Recommendation at the time of writing.

5.2.6. Web Workers

The W3C introduced **Web Workers** to bring the concept of concurrent running threads to the world of JavaScript[7]. This helps greatly in making pages more responsive while time-consuming functions are being executed. Background IO and shared tasks are also examples

[3] `http://www.w3.org/TR/IndexedDB/`
[4] `http://www.w3.org/TR/2010/WD-html5-20101019/dnd.html#dnd`
[5] `http://www.w3.org/html/wg/drafts/html/master/browsers.html#offline`
[6] `http://www.w3.org/TR/webmessaging/`
[7] `http://www.w3.org/TR/workers/`

for possible usage of Web Workers. The specification does not belong to the HTML5 standard itself but gets referenced by it [36][31].

5.2.7. Web Sockets

Web Sockets enable bidirectional real-time communication between a server and a client. This separately specified communication techniques might be used for example to update Web Storage data. Web Sockets can be used in an convenient way. Web Sockets allows a very low latency time because the connection will be kept open in contrast to Ajax-Requests [36][31].

5.2.8. Geolocation

The Geolocation API is also a separate specification referenced by the HTML5 specification. This API provides access to geolocation features easily, through a navigator object. In order to locate the device, many techniques are used on the client side. Apart from GPS positioning, that might not be available, network information, like cell information, or Wi-Fi hotspots in range can also be used for positioning [36][31].

5.3. CSS3

The new CSS3 standard brings a lot of improvements in styling and animation. They include transparency, CSS backgrounds and borders, shadows, CSS fonts, 2D/3D transformations, and animations, as well as transitions and transformations. One of the most important new features included with CSS3 are media queries. By using media queries a browser can check device preferences (height, width of the viewport, etc.). Based on this information, some styles or whole stylesheets can be chosen accordingly.

The WAP CSS extension defined by Open Mobile Alliance added attributes to CSS2 that should be used on mobile browsers. This attributes are all prefixed with a dash. This extensions for example contained Access key (`-wap-accesskey`), Marquee (`-wap-marquee`) and CSS form extensions (`-wap-input-format`, `-wap-input-required`) [31, 98] [11, 185-194].

5.4. HTML5 Support in Mobile Browsers

The status of many HTML5 APIs is still not final, but they are still being implemented in mobile browsers regardless. `mibilehtml5.org` offers a list with current implementation status of HTML5 features. A summary of these features can be seen in Table 5.1. Features that are important for the development of our e-Commerce app are highlighted in green.

Feature	**Safari on iOS**	**Chrome for Android**	**BlackBerry Browser**	**Internet Explorer (WP8)**
Web Storage	✓	✓	✓	partly
Multimedia	✓	✓	✓	✓
Canvas	✓	✓	✓	✓
Sockets	✓	✓	✓	✓
Workers	✓	✓	✓	✓
Geolocation	✓	✓	✓	✓
Form controls	✓	✓	✓	✓(only range)
Motion Sensors	✓	✓	only tablets	X
Network Information API	X	X	X	X

Table 5.1.: Overview of HTML5 implementation status.
Source: http://mobilehtml5.org

6. Development of Smartphone and Tablet Apps

"Learning a language nowadays is not very hard, they all look the same (literally). Really what you have to learn is the API." *Jean-Jacques Dubray, Founder, Convergence Modelin*

Knowing which platform to choose for the development of an app is crucial. Each brings its own unique pros and cons. This chapter briefly introduces a basic categorization and highlights the main differences between them. Many platforms use their own different programming language, but as Jean-Jacques said, the real hard part for developers is to learn how to deal with the platform's API.

Because we chose to develop the app with web technologies the presented frameworks that have been selected are representing the most promising ones in this category. The general prerequisites, design principles and usability factors that should be kept in mind when developing mobile apps are presented in Sections 6.6, 6.7, 6.8, and 6.9 of this chapter.

6.1. Categorization of Apps

The categorization of apps, from true native apps, to real browser based web-apps is essential. As the usage of these terms differs in literature, we will provide clarification about this categorization in more detail.

(HTML5) Web Apps: A web app is typically is a browser application that is accessible via its URL. But in the mobile context we, refer to a web application as one that seems to be running natively on the platform (like a native app) but is actually rendered inside a browser platform. This might be a web browser a web view or the OS itself in case of Firefox OS. These browser platforms offer additional platform integration or device features. On iOS, web apps can provide a full screen experience by using special meta tags that will hide the browsers navigation elements.

Another special kind of web apps are platform specific packaged apps: These apps are web apps that are distributed inside a predefined packaged. Examples for this kind include Chrome Apps, Firefox OS apps, Symbian standalone web apps and others. Their package contains all necessary files and a description file (manifest) that stores additional metadata. These apps need to be installed via the browser and many platform providers also offer app stores to find and download these apps. They also do not rely on a hosting server, so they are conceptually very similar to native apps [31][52, 148].

Hybrid Apps: Some problems for web-apps like missing platform features can be filled via the usage of wrapper frameworks. A well-known one that provides such a wrapping is

Apache Cordova with its PhoneGap implementation from Adobe. This way a pure web app can be turned into a native wrapped app. The content will be shown in a WebView and additional native features are available through the framework. This includes native events, reading/writing contacts, accelerometer, filesystem support and others. The main application itself is still developed using web technologies, but through the native wrapping it is possibility to distribute it via the native app store

Pure native apps: For a pure native app, all the visible views are developed using native elements (e.g. Buttons, ListViews, ImageViews, etc.). The app itself (business logic) is also developed in native code, but of course some content (text, images, etc.) might be fetched from a webservice [37, 30].

The most essential thing to consider when developing any kind of software is knowing your target user and their core expectations. The available budget should also be clear. In Chapter 4 we already introduced some basic knowledge about platforms, app stores, and browsers for the mobile context. We also highlighted their market distribution. With this information in mind, we now need to know about the different development possibilities. This chapter gives an introduction to native app development, mobile web-app development, and the development of hybrid apps. Pros and cons for each of them are summarized in Section 6.4.

The following section will focus on the question: Which type of app(s) should be developed, and what consequences will this decision have? Thus, details on native iOS and Android development (Section 6.2) as well as mobile web app development Section 6.3) are highlighted.

6.2. Native App Development

Each mobile platform has its own application development framework or API to develop apps. These frameworks are built on top of the underlying Operating System (OS) and are optimized to distinct hardware models and therefore they offer the full spectrum of possible device features, graphic acceleration and seamless integration into OS functionality.

iOS development To develop native iOS applications the *Cocoa Touch* application framework is used. The programming language for development is Objective-C. The Xcode IDE is available for developers (using Mac OS computers) and contains several tools for development. Included is a device simulator, a debugger, and a UI-designer (Interface Builder), etc. Testing on the simulator is fine, but it is obviously not possible to test device functionality like SMS-messaging. To do so you need a real hardware device and a signed provisioning profile. There can be maximum 100 devices in use for testing purposes and they all have to be provisioned with their device-id in the first place. The end customer distribution takes place via Apples App Store.

Android development For Android development Google provides the Android SDK. Coding takes place in an adopted Java version that runs on a Dalvik VM. The second but not very likely way is the Android NDK where developers can write code in C/C++ to gain a higher performance. The intended IDE is Eclipse with an additional plugin or the Android Studio IDE. Google provides emulators, a debugger and a graphical UI-designer. On device testing

and debugging can be easily achieved by allowing third party software and enabling the debug-mode in the preferences of the device.

For the mainstream consumer market, only iOS and Android have much relevance (see Chapter 4 for market distribution). Depending on the purpose of an app, its target user groups, or target country other platforms might be considered as well. For the purpose of our e-Commerce app, we are going to focus on the aforementioned two platforms. Table 6.1 additionally contains information about BlackBerry OS and Microsoft's Windows Phone [53][54][55][56].

Platform	**Type**	**Language**	**API**
Android	Open Source	Java for Android	Android SDK/NDK
iOS	Proprietary	Objective-C	Cocoa Touch
BlackBerry	Proprietary	Java	Java/BlackBerry API
Windows Phone	Proprietary	C++/C#	Windows Phone SDK

Table 6.1.: Native Development Overview

Table 6.1 shows primary development languages for popular mobile platforms and their development SDKs offered by platform vendors.

6.3. Mobile Web App Development

"It's Not the Mobile Web; It's Just the Web!" writes Maximiliano Firtman the Author of *Programming the Mobile Web* [31]. He wanted to make the point that it is the same web we access from a mobile device or a desktop computer; only the visual presentation might be different. When developing or optimizing for smartphone and tablet devices, we have to address the different screen sizes, input techniques, limitations in bandwidth etc. But in order to emphasize that we are developing for small screen devices, we often use the term mobile web.

The development is similar to web development that targets desktop sites. Technologies like HTML5, CSS3 and JavaScript, as explained in the last chapter are core technologies for modern mobile web development. Some aspects, like low bandwidth, offline support, or the geolocation feature are obviously playing a more important role when developing for mobile devices.

Differences to native app development:
Unlike native code, web app projects share (nearly) all code among platforms which is also one of the big advantages of web apps [57]. Exceptions here are for example platform specific JavaScript code segments. Optimizations for different platforms might happen in there. Also platform specific stylesheets are possible and can realize an adopted user interface for each platform. A big advantage for web-developers is that existing knowledge, tools and processes for can be re-used. But using web technologies means also a higher grade of abstraction and therefore results in performance loss. This is getting less important if hardware capabilities keep on improving, but at the time of writing there is still a notable gap between native app

performance and web app performance [31][11]. Reasons for this have already been outlined in Section 4.1.1.

Along with the performance, the distribution and deployment of mobile web-apps also differs from workflows for native ones. On the one hand, access to direct and instant publishing is a huge advantage for web-apps. There are no restrictions and no approval processes there to delay the deployment procedure. But on the other hand, they cannot be distributed via conventional platform-specific app stores without being wrapped inside an application package. That is why new app stores for mobile web-apps emerged [1] [2] [3], but their size is not comparable to those of native app stores [34].

6.3.1. Frameworks

Modern frameworks and libraries can improve the development process and thus the outcome of mobile web app development. The most influencing ones at the time of writing are jQuery mobile that is based on jQuery and Sencha Touch based on Ext JS. They are both designed for mobile web development but as we will explain they follow very different approaches to fulfil that. The following section will explain the advantages and disadvantages when choosing between those two competitors.

Sencha Touch

Sencha Touch is developed by Sencha Inc. The company was founded in 2008 and is backed by big investors like Sequoia Capital, Radar Partners or JAFCO Ventures.

They claim to have 2+ Million developers around the globe using Sencha technology and 5+ Million product downloads. Over 70 worldwide meetup groups and 25+ service partners underline their effort in maintaining a big developer base [4]. Frameworks and tools for web developers are central product categories provided by Sencha. This includes Ext JS HTML5/JavaScript framework for developing dynamic desktop sites and Sencha Touch. By offering free of charge commercial and an open source licenses for the Sencha Touch SDK barriers for new developers are very low.

A big developer base and an active community are important aspects when choosing a framework. But the availability of high quality learning and training materials as well as sample applications are also vital. To address these concerns, Sencha offers a learning center with tutorials and a full API documentation. The latter is covers all API elements but sometimes offers very limited information. More real-life code examples would have been helpful in some cases. But of course, not everything they offer can be made free of charge, thus, Sencha Care is a paid support subscription that offers professional development companies advanced support options and also access to additional services offered by Sencha. Premium

[1] `http://openappmkt.com/`
[2] `http://www.zeewe.com/zeewe/web/`
[3] `http://www.apple.com/webapps/`
[4] http://www.sencha.com/company/

support customers get a faster response time for support requests, and can receive support by Sencha product experts by way of code reviews, performance tuning, or remote troubleshooting. In addition, Sencha Services can be acquired. They offer UI design and development, system architecture specification and review, custom component development, and comprehensive training programs.

Sencha Touch targets Webkit based browsers and thus runs on the Android/Chrome, Safari and the BlackBerry browsers. Their widgets offer native styled UI elements (based on iOS design). Our experiences showed a high-performance app on iOS devices, but significantly lower performance on (latest) Android devices [58][59][60].

Features

- Application Framework
- UI Widgets
- Built-in Gesture Library
- Sencha Commands (includes a native build tool)
- Custom Scroller
- Routing and History support

Problems

Sencha Touch is known to have a long learning curve. As previously mentioned, the performance is very good on iOS devices, but poor on Android. Even fairly new devices like the Samsung Galaxy Tab II offered a bad scrolling experience. One explanation for that is the stock Android browser, as discussed in Section 4.3 is lacking in the appropriate hardware acceleration. The predecessor, Chrome for Android, however, demonstrates a superior performance.

Development

Coding takes place in JavaScript, HTML (inline) and CSS/SASS. So basically every editor might be used for development. Sencha offers a Plug-in for the well-known Eclipse IDE that supports code completion and code assistance. In addition to the Sencha Touch SDK, a local Web Server would be advantageous to give devices as well as other team-members the possibility to access the codebase. Apache is suggested for this. Sencha officially suggests the Google Chrome browser, or Apple's Safari browser. Both offer very convenient possibilities for debugging. Additionally, it is advisable to have native SDKs (including Xcode as well as Android SDK) and development tools installed in order to utilize simulators and deploy apps to native packages. The widely used Model View Controller (MVC) pattern comes into play to develop maintainable and reusable code. Separation of presentation and business logic is state-of-the-art for modern web and software development. Client side MVC brings relevant advantages. Better, more modular development for big projects, easier understandability, improved options to share code across applications, improved testability and reduced test effort for new development [59] are some of them [59]. Controllers, views, stores and models are typically located in respective sub folders [5] inside the project.

[5] http://www.sencha.com/blog/architecting-your-app-with-sencha-touch-2-mvc

One of the reasons for a longer learning curve is most likely the complex but also powerful Sencha class system. Thus, the framework is very well equipped for advanced developers, familiar with big JavaScript projects and best practices [6]. The Sencha class system is based on the Ext JS code. It provides inheritance, on demand dependencies, mixins, config options and other useful concepts. Beside the typical MVC elements, profiles can also be defined by customizing the app appearance for certain device types [7] [59].

Several managers offer layout capabilities that enable to adopt the positioning of UI-widgets on the screen. The widgets are designed to look like native UI-elements (mainly iOS) but can be easily changed in appearance as well. Therefore (Syntactically Awesome Style Sheets (SASS) is used to change a generated Cascading Style Sheets (CSS) file. These User Interface (UI) elements are called *Components* and might be contained inside *Containers*. Their appearance can be defined in HTML inside a view or during instantiation. Layouts open up the ability to position Components inside a Container. Vertical and horizontal layout, card layout or the simple fit layout are predefined. Also docking of Components is possible[8]. The appearance of views is also defined inside JavaScript (with inline HTML code). This offers more flexibility.

Another core part for this framework are events that can be fired by components. Listeners are defined to perform actions when these events occur. This also helps to keep a loose coupling between different modules inside the code.

Sencha includes a custom scroller that uses hardware-accelerated CSS3 transitions to provide a smooth scrolling experience. Furthermore it offers a *Gestures Library* to manage tap, pinch and swipe inputs.

Sencha Commands is a command line tool to provide the possibility to conveniently set up the initial project structure, generate MVC components and even build or deploy your app. This is possible across multiple platforms, but of course basic restrictions cannot be circumvented. Thus native packaging is only possible within Mac OS and compatible hardware [9].

In addition, Sencha Touch SDK and the Sencha Commands also offers [58] [59]:

- The proprietary Sencha Architect, a design tool that assists building the UI for new apps (mobile and desktop) inside an easy drag-and-drop application.
- Sencha Animator for easy CSS3 animation editing for mobile platforms.
- Sencha.io to provide user data management, messaging and deployment in the cloud.

jQuery Mobile

As its name implies jQuery Mobile is based on jQuery - the very widely used library for easy and effective JavaScript development. A huge community stands behind these two libraries that are available under the GPL or MIT License. They are known to be easy to learn and very convenient to use. The first Alpha version of jQuery

[6] http://docs.sencha.com/touch/2-0/#!/guide/apps_intro
[7] http://docs.sencha.com/touch/2-0/#!/guide/getting_started
[8] http://docs.sencha.com/touch/2-0/#!/guide/layouts
[9] http://docs.sencha.com/touch/2-0/#!/guide/command

Mobile was released in October 2010. Mobile industry leaders like BrowserStack, Bocoup, BNOTIONS are supporting the framework.

The main focus of jQuery Mobile is to provide functionality to build interactive, rich, and theme able user interfaces that are optimized to work for all popular mobile devices (phones and tablets). Thus, no application layer functionalities are provided. Therefore either an additional framework can be used (like Backbone.js or Knockout.js) or a custom implementation of any suitable pattern might be developed. This leaves more flexibility for projects that are not suitable for a predefined architecture. Nevertheless it might also mean a bigger initial effort. All jQuery Mobile enables is done by using only HTML5 standard code. In contrast to Sencha, jQuery Mobile supports a vast amount of different browsers. Their website[61] provides a full list of supported platforms [62][46].

Features

Main features and advantages of jQuery mobile[46, 15]:

1. Rich & Interactive Apps
2. Lightweight
3. Optimized for Touch Devices
4. Easy to learn
5. Plug-ins Available
6. Events for Interaction

Problems

jQuery Mobile is not as performance-capable as Sencha Touch. Unlike Sencha Touch there is no native building included. But the framework is PhoneGap friendly and therefore the external wrapper framework can be used to build a native version of the app. The documentation is very limited but there is a great support through its community.

Development

To start development any suitable editor can be used and any browser should be able to display the web-app. Also worth mentioning is that Adobe Dreamweaver has official support for jQuery Mobile.

To start development the framework needs to be included inside the website. This is typically done by referencing it via an Content Delivery Network (CDN) inside the documents `<head>` element or by locally including it to the project.

A view is defined in HTML5 and nearly no JavaScript is necessary for that. However, jQuery Mobile is easier to start with, if you are not an experienced JavaScript developer.

To select specific elements by, class, type or id the jQuery selector ($(...)) is used. This kind of syntax is a combination of JavaScript and CSS syntax, which is very intuitive to use. Interactions on the website are handled by events. Unlike in standard JavaScript, they are absolutely reliable across browsers. Custom events can be triggered to extend the functionality

of the basic jQuery framework. Events can be easily bound to an jQuery object by just defining a function to be executed for a given event [62, 10].

As already mentioned jQuery Mobile supports many different platforms and browsers to present a uniform appearance across all of them. For legacy platforms with missing APIs the framework provides a fallback mechanism to ensure a basic functionality of the app. In the way of *Progressive Enhancement* additional functionality is provided on advanced platforms [46, 17]. jQuery offers a lot of extensions developed by the jQuery Mobile community. It is easy to build new extensions and many developers are willing to give back share their code.

The architecture of jQuery Mobile is very simple and mainly focusing on the presentation of content. Roles are used to define how elements should behave. Therefore the `data-role` attribute is used. There are many predefined roles like page, header, navbar, button etc.

Another important feature of jQuery Mobile is the ability to support theming. Thus, the appearance of the whole app can be easily changed to another set of styles. On all A-grade browsers jQuery Mobile offers: - Easy Ajax APIs for asynchronous requests. - A hardware accelerated page transition feature. - Many different animated transition between views.

So jQuery Mobile offers a convenient way to UI develop for smartphone and tablet devices. Additionally it is suggested to use a JavaScript MVC application framework. This helps to keep the code more flexible, maintainable and reusable for larger projects [62][46].

6.4. Comparison Web vs. Native Apps

Beside development related differences between web apps and native apps, we also want to outline basic differences that will particularly affect the end-user. One of the first significant differences a user will determine is the way how he finds and accesses the app. Native apps are typically provided via an app store and need to be downloaded an installed prior usage. Web apps are simply accessible like normal websites through their respective URL and the code as well as the content is downloaded on demand. Each approach has advantages and disadvantages. Web apps are per se easier to be updated because every change on the server side will immediately affect all accessing users. This advantage is lost if web apps are packaged and distributed via app stores. A major and in literature often underestimated advantage of web-apps is the accessibility of app screens via particular URLs. This enables search ability and accessible through search engines (for all content, not only the app itself) and the possibility for advanced social integrations such as Facebook "likes". This is particularly important for e-commerce applications. When we take a look at performance characteristics, the available number of platform specific features and user interface characteristics, native apps are definitely at an advantage over their web app counterparts. Native code can be executed only on one particular platform, which also results in a performance benefit. Also a native look and feel through well-known UI elements and platform specific navigation methods are advantages of native apps. The appearance of native UI elements can of course be mimicked (Libraries like Kendo UI claim to provide such possibilities across all major platforms) but it is not the case for a standard web app implementation and in many cases not intended. While HTML5 is offering an increasing number of mobile-specific features it will never provide all available possibilities a native SDK can provide [5][57][63].

A brief overview of advantages and disadvantages between native app development and web-app development is shown in Table 6.2[22, page 15-16][57][35][31].

Native		**Web**	
+	Best possible performance	-	Performance not as good as for native apps
+	Platform specific appearance	-	(Typically) a one fits all solution
+	Offers all possible platform features	-	A cut set of platform features is available (hybrid apps are extendible via native code)
-/+	Distributable via native app stores	+/-	Available through a web url (and app stores)
-	Deployment is not quickly applicable (especially on iOS)	+	Changes on the server code provides instantly updated client apps
-	Expensive to develop/maintain for multiple platforms; requires distinct knowledge/specialists for each platform	+	Cheaper to develop for multiple platforms; less platform specific knowledge is needed

Table 6.2.: Comparison of Native Apps and Web-Apps

The following questions should be considered to decide which kind of app is more suitable. Luke Wroblewski [63] writes in his book *Mobile First*, that there are also good reasons for doing both. As pointed out in Table 6.2 it is a fact that native apps provide some advantages in contrast to web based apps. Reasons for this are that native code is of course optimized to run on the underlying platform. But nevertheless there is one essential advantage that makes web apps so important. They are reachable via a simple URIs, without the need to install anything on my device. Additionally it is possible to provide a link to a specific content view inside the app which is an essential and very powerful core concept of the web. This allows people to find and share information. So it is obvious that Wroblewski defines the access as the biggest user benefit for a mobile web experience.

Which kind of apps are suited to be developed using web technologies?
Certain categories of apps are better suited to be native apps than others. Apps for enterprise usage are per se very different from consumer-focused apps. Gartner stated in a press release [64], they are predicting that 90% of B2E apps will be either web or hybrid solutions in 2015. In contrast to that they estimate "only" 60% web/hybrid apps in the consumer sector. So Games and rich media apps with a lot of interaction and animations are tend to be implemented as native apps. Of course as like in every context exceptions prove the rule. Apps for mobile banking solutions, e-Commerce apps and services that need to be accessible with as many different kind of devices as possible are better served with a web solution. This data driven applications do not necessarily need fancy animations or access to a lot of device specific features.

6.4.1. Differences in Costs

The most important influence on a Software project is of course the available budget.

Due to the aforementioned cross-platform capabilities of web-apps, they are cheaper to develop for multi-platform solutions. But beside the initial investment, even more important will be the costs for ongoing maintenance and improvements. High maintenance costs for multiple platforms can quickly make a service prohibitive.

Data driven and content focused apps should most likely start with a web-app solution. Native apps for various platforms might follow after some time.

Apps with the need for a highly polished and snappy user interface will be well served by starting with a single platform or in some cases with an Android/iOS native implementation.

How should the app be distributed?
The important question here is whether the app should be only distributed via app stores or if it must be also accessible via a URL. The ability to run inside a browser window might be necessary or eventually brings big advantages. In this case it depends on the budget if only a mobile web version or also additional native ones are feasible.

6.4.2. Reasons Against Web App Development

One point that was mentioned when companies switch from HTML5 apps to pure native apps is performance. Although performance of web based apps is getting better they will always be less performance-capable than native ones [63]. The second reason for a switch to native app development is tooling support. Kiran Prasad, LinkedIn's senior director for mobile engineering outlined in an interview with VentureBeat[10] the they mainly dropped HTML5 because of a lack of good performance analysis and debugging tools. He claims that there are tools out there, but they are simply not comparable to their native counterparts.

The stated advises only reflects general suggestions by sighted sources [11, 34][5][57][63]. Nevertheless each single app is as unique as its people who are developing and distributing it. Therefor each decision should be made based on these unique circumstances. The environment is changing very fast. E.g. Brian Fling [11, 146-150] mentioned in 2009 several reasons to build native apps. They included: charging for it, using specific locations, using cameras, using accelerometers, accessing the filesystem or offline usage. But this is not correct any more. Some of those features are already possible in modern mobile web-platforms as Chapter 5 showed. The access to more advanced hardware features and native building of mobile web apps is certainly possible by using wrapper frameworks.

6.5. Developing for Mobile Platforms

The following sections will provide a collection of design principles, best practices and user experience guidelines provided by various sources. Some of them will be related to a certain platform, while others are applicable for all platforms including web apps.

[10] http://venturebeat.com/2013/04/17/linkedin-mobile-web-breakup/

6.5.1. Web App Principles

When developing web apps (not only for mobile devices but in general) the following principles might be considered[11], [12] [63]. Also the *Desing Guide* provided by Mozilla on how to build mobile web apps for Firefox OS apps provides very helpful information. Those suggested principles are explained in more detail in Section 6.7 and Section 6.8.

- Designing for mobile first
- Build single page apps (not applicable in full extent for all kind of apps)
- Use a REST API
- Design for a great UX
- Use W3C standard web technologies
- Optimize for touch interfaces
- Be consistent
- Provide immediate feedback
- Aim for a pleasant first launch experience
- Present action choices if possible
- Be concise
- Prepare for interrupts
- Minimize typing
- Use appropriate language

6.5.2. Considering Platform Related Differences

An app should try to follow platform specific appearance and paradigm as good as possible. With respect to the most dominant platforms, only iOS and Android related differences have been investigated here. The Android design guide [65], the Android development guide [66] respectively as well as the iOS human interface guidelines [67] provided background information on the following findings:

Back navigation patterns: Apps for the iOS platform offer an on back navigation button in the left corner of the navigation bar. The button should either be named *back* or contains the name of the previous screen. Android offers two options. There is the *physical*[13] back button that works on every screen. Additionally there is the (optional) up button placed in the left corner of the action bar. The up button takes the user one step up inside the application hierarchy.

Tab navigation: Android tabs are typically placed on the top of the screen. There can be placed any number of tabs by making the tab bar scrollable. On iOS the location of tabs need to be at the bottom of the screen, and it does not allow for more than 5 tabs to be displayed at a time.

Switching screens: On android devices apps should make switching between different screens available through the control spinner in the navigation bar. On the other hand iOS uses the concept of page controls that are displayed as divided segments in the top bar. Beside this traditional concepts both platforms also provide the ability to use an *off the canvas* approach to switch between main views. This concept is explained in Section 6.8 [68].

[11] `https://blogs.atlassian.com/2012/01/modern-principles-in-web-development/`
[12] `https://developer.apple.com/library/safari/technotes/tn2010/tn2262/index.html`
[13] Not always literally physical. On some devices the back button is placed as on screen element in the lower part of the display.

Beside this also the animation between screens is different. While iOS strictly enforces the left-to-right flowing animations for forward actions and left-to-right for backward navigation. Android on the other side leaves it up to the developer how transitions are animated, although a consistent and reasonable type of animations is advised.

In app actions: Actions inside an app can be made available to users in different ways. Android‘s preferred method is to place them as action item into the action bar. Additionally, a split action bar that appears at the bottom of the screen can be used to place action icons. To provide more flexibility the items inside the action bar can change due to the users context [14] [68].

Screen resolution and sizes: Android devices are available in various sizes and resolutions. The huge fragmentation brought up some concepts to differentiate classes of devices. First of all there is the "density bucket" categorization. Here devices are categorized by [15]:
Screen size: small, normal, large, and extra large.
Screen density: ldpi, mdpi, hdpi, xhdpi.

Apples iOS devices are available in two sizes and 3 different densities. Thus the visual screen design does not need to be as flexible as Android design [68].

6.6. The Mobile Context

A sentence by Rachel Hinman [5] sums it up in a few words: "Mobile context = anywhere and everywhere". This makes the biggest difference to the previously known static context of desktop applications. But how can you design for unpredictable situations where an app might be used?

First of all, interruptions during usage will happen. Much of mobile usages is simply done to pass time. Thus, the power to enable the user to continue where he/she left of is a great feature to help avoid unnecessary actions. The attention of the user might also be disturbed by noise or other influences, so any reduction of cognitive load is a very important point for all small screen designs. The basic rule is to reduce about 80% of the information/content that would be visible on a desktop counterpart (website, software). From an economical point of view, the opportunity costs need to be as low as possible. This can for example mean to provide very effective and quick solutions for the most important usecases of your app. Of course the opportunity cost may vary depending on the users context, however a good baseline is always to keep it as clear and simple as possible [5][31, 80].

When developing for mobile devices it is critical to keep in mind that the users‘ context is very different from a desktop user.

The users context includes [31, 74]:

- Where is the user? (Position)
- What is he doing at the moment? (Walking, driving, sitting at home)

[14] `http://developer.android.com/guide/topics/ui/actionbar.html`
[15] `http://developer.android.com/guide/practices/screens_support.html`

- What is his goal? (Use cases)
- Can any mobile device features be helpful for the user to achieve this goal? (GPS, Barcode Scanner, ...)

6.7. Design Principles

These basic design principles define some guidelines for interaction designers. They contain no recipes how an interface should look like, but provide guidelines to create an UI with a better user experience.

There are two main objectives [69]:

- Optimizing human performance.
- Optimizing user satisfaction.

In general these guidelines reflect common sense knowledge of experienced interface designers. When applying these rules to a real world application, often they are influencing one another. Thus a good overall experience can often only be provided with some kind of trade-off. Because each app and each user interface is unique, they cannot be applied to their full extent in every case. Sometimes little but carefully chosen exceptions may also result in an outstanding experience. The design principles will provide a foundation for user experience considerations that follow afterwards.

Visibility Functionalities as well as the current state of the system should be visible to the user. Core features should be easily accessible in every view [70, page 26]. Examples for this principle include the possibility to go back to the previous view (back button), or the action bar on Android to provide access the most important actions at all times.

Feedback Immediate feedback, as we normally get it in our everyday lives is very important. Users should instantly be informed if their action resulted in any kind of state change. Therefore it is crucial to generate appropriate feedback. Choosing between audio, tactile, verbal, visual or multiple kinds depends on the application.

The reaction time of the app is very relevant for the usability and user experience. This means that any kind of feedback should be presented to the user within a certain timeframe. The faster the better. A reaction time of 3 seconds or more is not acceptable to the average user [52, 153]. For network actions it is a possible solution to show a loading animation to provide instant feedback. As long as there is instant feedback indicating that there is still data to be fetched, loading times are not that critical any more [70, page 25-29].

Constraints Showing only permissible interaction possibilities to the user is one example for constraining. This principle prevents the user from useless interactions [70, page 25-29]. In desktop applications this principle is often used for menu items. When a specific item is greyed out, it indicates that it is not an accessible action.

Consistency Following a consistent approach to fulfill similar tasks across an application. This helps the user to apply his knowledge across different parts of your application. Research in [48] showed that even popular apps like the *Popular Science* Magazine app or the *USA Today*

app are showing lacks of consistency. Unpredictable behaviour results in a lower usability and can even lead to a frustrated user. Using platform specific guidelines to provide consistent interaction techniques across all apps is the best approach. This way the user does not have to "learn" how to use each new app he downloads.

For web-apps it is often not possible to adopt the appearance in any platform-specific way. Therefore different solutions might be chosen here [48, 146-150][70, page 25-29].

- Adopt for the most important platforms (iOS, Android)
- Provide the best possible option that fits all platforms

Affordance Affordance means to follow interactions of the real world (button → push, doorhandle → pull) that are made obvious by physical constraints. Affordance in terms of user experience design stands for ways on how to make interaction with an interface obvious in a specific way [70, page 25-29]. An interface designer has to avoid low-affordance tap areas. Users are not willing to try out touching every part of the screen, to see if something is happening or not [48, 138-140]. Affordance does not necessarily mean that design elements should look like real world objects (e.g. real looking knobs, ...), but their design should lead the user to the designated interaction (push, drag, ...).

6.8. Smartphone and Tablet Usability

In addition to the previously defined general usability principles, there are further ones specifically for tablet and smartphone apps. Mobile operating system vendors offer guidelines for interface and user interaction design (Android[16], iOS [17]). Some of those guidelines are the same or similar on various platforms and other are platform specific. The following issues and principles reflect only those suitable for Android and iOS devices.

Connectivity Problems Design the app to be able to handle changing connection speeds and suddenly interrupted connection [52, 151]. The loss of the system state or missing data is a very annoying issue for the user.

Avoid Typing as much as Possible: Typing on mobile devices is difficult in comparison to desktops. Therefore avoid the usage of long form inputs. Possible solutions for that issue include [48]:

- Provide defaults and suggestions
- Prefer spinners over text fields
- Use GPS, voice input or other alternative input mechanisms when appropriate
- Only ask for data that is absolutely necessary
- Support a *Copy and Paste* mechanism

[16] `http://developer.android.com/design/downloads/index.html`
[17] `http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html`

Low Interaction Coefficient The most used functionalities in an app should be accessible in a few steps [52, 153-154]. This is very important for small screen devices, because any difficult interactions with many steps might overcharge the user.

Avoiding the Fat Finger Problem A finger used as input device as successor of the mouse pointer also brings new issues. Mainly because it is not as precise as a mouse pointer. This means that buttons, links and other tap areas have to have a certain minimum size to be reliably clickable. Also the distances between selectable areas need to have specific values. Therefore the platform specific user interface and design guidelines define minimum sizes for menu items as well as spaces between them.

Less is More Nielson describes the main point of mobile usability as "focus users attention on the essential content. [48, 101]. This is a very important aspect for smartphone and tablet apps. Researches [71] also showed that this depends on the application type. E.g. for e-Commerce apps it is problematic to cut out information or features.

Reading and understanding complex content on a small screen is much harder, than it is on a desktop screen. Only a part of the information is visible at a time and users have to remember the rest of it. This so called peephole effect results in 108% decreased understanding, when reading on a mobile screen [48, 102]. In contrast, Cloze tests (empirical comprehension test) showed 60% higher scores for desktop reading against reading on mobile screen. Additionally reading on a smartphone requires a lot of scrolling and navigation, resulting in more time to read and understand the whole text [48, 101-122].

To minimize this problem, remove as much content as possible until only the most essential portions remain. Rephrasing any text (e.g. only short sentences and well formatted, concise writing style) might be advisable. A very powerful and widely used approach is to move secondary content to additional views that are only shown if the users explicitly request them.

Low Memorability and Gestures Gestures are an intuitive way of interaction. Most users are only familiar with the very basic ones like tapping, dragging and swiping. Other, more advanced actions, like double tapping or touch and hold are not as common, so using only a few well known gestures is generally the best practice, although the basic functionality should not rely on knowing them [48, 141-143].

Linear Paging In many cases, it is good practice to let the user scroll through a longer episode of text. Linear Paging means that the content is divided into pieces (like pages, or cards) that are separately accessible. Cards, for example, provide a screen filling fixed-size design, and as a result, fancier visuals and beautiful layouts are possible, but should still only be used in special cases [48, 153]:

- On Tablets, paging might be used if the view represents a book. Usage of swipe gestures for page navigation and preloading of content is suggested.
- Usecases that require wizard-style navigation are also well suited for paging. An example would be a checkout for an online shop, were steps should be viewed one after the other.

Alphabetical Sorting This kind of sorting is usually a bad choice. It is only relevant if the most likely use case is a user looking for something with a name he knows (state, first name, etc.) [48, 123-129].

Scrolling Scrolling should be limited to one direction. Simultaneous vertical and horizontal scrolling is considered bad practice [52, 155]. The decision of whether or not to use horizontal or vertical scrolling, once again, depends on the app type, but should be made visible through small hints in the design. Arrows, ellipses at the bottom of a page, or partially visible content on screen edges can indicate horizontal/vertical navigation. It is important to stay consistent across an app [48, 153].

Too Much Navigation Too much navigational content at once results in apps that are difficult to use. This is reflected in the aforementioned "less is more" approach for the content of a screen. In addition to this rule, a UI should only use a few different kinds of navigation techniques[48, 159].

Avoid Orientation Problems Many apps provide landscape and portrait views. Although this is very convenient for the user it may result in additional problems like[18]:

- Unavailable navigation elements.
- Inconsistent behavior in landscape and portrait mode.
- Incorrect positioning of navigation elements.

Off the Canvas Pattern This is a very new approach for offering navigation elements. By default the navigation elements are hidden outside the screen area until the navigation button is pressed. Now the menu swipes in from the side. Swipe gestures can be used in addition to the navigation button. This pattern is also known as a "navigation drawer" or "sidebar layout" [52]. It offers a central method of navigation through an app, without taking up too much onscreen space.

6.8.1. Tablet Usability

In terms of usability the difference between smartphones and tablets seem to be quite small. Although there are still some differences that sometimes turn out to have a big influence.

As we already mentioned in Section 4.4.2 the only obvious difference on the first impression is just the size. That is why most of the design recommendations are very close to the smartphone ones. Nielsen et al. [48] investigated the usability of the iPad and the Kindle Fire. They found the following core differences for these devices in contrast to typical smartphone usages:

- Reduced mobility (see Section 4.4.2)
- Full websites work much better than on smartphones
- A tablet often is a shared device
- Usecases, frequency of use and session length are different to smartphone usage

[18] http://uxdesign.smashingmagazine.com/2012/08/10/designing-device-orientation-portrait-landscape/

Thus it is important to consider the changed context of use when designing for a tablet. The usability and user experience of a m-Commerce application are critical. Today's customers not only expect a well-designed user interface. Their expectations in a smooth and intuitive interaction and engaging functionalities are high.

Accidental Activation A combination of the "fat finger problem" and the visibility design principle come into play on tablet devices. A larger screen requests wider movement, in combination with to small touch areas or too little space between them, the user can occasionally activate UI element unintentionally. Especially true for apps without an undo or true back button, this can lead to situations where the app is caught in an unwanted state [48, 143-145].

The Print Metaphor On tablet devices, the linear paging approach for books and magazines, in combination with swipe gestures, is a very intuitive one. But this type of navigation alone can be very frustrating for users who are looking for a single article, it is recommended to offer additional possibilities via direct links etc. when using this gesture based technique.

Another problem that might occur is the interface with an image carousel, as it also represents some kind of (horizontal) scrolling element inside the page. This can lead to confusion if the carousel is too large. To reduce this possibility, the editor might leave the corner areas free of those components, thus allowing swipe gestures starting in these areas [48, 151-159].

Tablet Web is between Desktop and Smartphone Web apps and websites for tablet devices are typically designed more like the desktop version rather than the smartphone version. That is why Apple suggests developers to serve the desktop version instead of the smartphone version if no dedicated tablet version is available[19].

6.9. Design and Constraints

Conventional websites are very inconvenient to use on a small screen device, since a good deal of zooming and scrolling is required, making the usability factor very poor. Studies shown here indicate this as a reason for users preferring to use apps instead. Even websites optimized for mobile platforms tend to have an inferior user-experience to a native app [48, 132-133]. But why? The following section will attempt to answer this question, and provide more insights on user-friendly app design.

6.9.1. Mobile Design

Good Mobile Experiences Progressively Reveal their Nature

It is not enough to simply create a minimized version of a desktop application or a website. A whole new conceptual and psychological model needs to be considered. The constraints defined in Section 6.9 inspired new kind of interaction designs. The user experience on small-screen devices is an explorative one; a consumer acquires more insight with each new screen. Interfaces, in general, should offer fewer possibilities at a time, with any information presented to the user truncated and divided into reasonable portions.

[19] `https://developer.apple.com/library/safari/technotes/tn2010/tn2262/index.html`

The Nested Doll pattern: Views are offering more and more detailed information the deeper the user dives into [5].

Hub and Spoke pattern: The "Hub and Spoke Pattern" defines a navigation method where users often must return to a central screen (like an airport hub). This pattern is suitable for offering many alternative functionalities with a large number of different views [5].

Bento Box pattern: A pattern that is well-suited to tablet interfaces. In the style of a Japanese bento box, related information is separated into specific areas on the screen. An example for this pattern would be the live tiles in Windows (Phone) 8 [5].

The Filtered View pattern: The same information, filtered and presented in different ways, hence the name. Handling large sets of media data is ideal for this pattern [5].

The Cloud and Applications as Natural Set Points for Mobile Experiences

Using the cloud as a set point for the development of mobile apps, this offers huge possibilities to access, manage, and share large amounts of data. Besides relieving the burden of storing all data locally, it also brings convenient social interaction possibilities. Good examples of this approach are: Dropbox, Google Drive, and Netflix. The ability to access your data from anywhere, with any kind of device is very convenient, but a downside of these services is their heavy reliance on internet connectivity [5].

Content Becomes the Interface

UI designs today move away from unnecessary graphical ornament and effects, like color flow and drop shadows popularized in early years of the web. The designs of the present are simple, clean, and smart. This approach is characterized in the Windows (Phone) 8 user interface [5].

Use Uniquely Mobile Input Mechanisms

Leverage new intuitive input techniques and device features [5]:

- Touch gestures for navigation
- Accelerometer to recognize device shaking
- Speech recognition
- Bar-code and a QR-code scanning
- Location service(wireless-networks, GPS)
- Short range communication: NFC, Bluetooth

New Interaction Models

Beside task oriented apps, new kind of motivation driven apps appeared [5]:

- Interactions accrue Value over Time. Examples are social network apps like Facebook, twitter and co are good examples. User normally do not open the app to fulfill a specific task.
- Interactions that Facilitate Exploration and Discovery
- Interactions that detect the user's intent by way of sensors. This is a very intuitive way of making software more engaging and effective. Using orientation changes to adopt the interface is one example. Showing the Google maps area around the user's current position is another.

6.9.2. Mobile Device Constraints

The following table contains constraints that have to be kept in mind when developing for smartphone and tablet devices.

Mobile Device Constraints	**Mobile Environment Constraints**	**Mobile Human Constraints**
Small form factor	Can be used in almost any environment.	Easy to lose.
T9 and/or QWERTY alphanumeric input	Can be stored in almost any environment.	Language and metaphors aren't always appropriate for some cultures.
Small screen	Screen glare can occur in bright sunlight.	Focus required for cognition.
Camera/video	Sensitive to wear and tear (dropping in water, dust, etc.).	Alphanumeric input is challenging.
Battery operated	Use is prohibited in some environments.	Socially awkward or unacceptable to use in some circumstances.
Cellular, WiFi, and Bluetooth network access	Unable to use because of lack of network connectivity.	Small text and graphics are sometimes difficult to read.
Microphone	Electricity is scarce in some environments.	Difficult to hear in some contexts.
Speaker	Financial constraints are based on location (roaming charges, data costs).	Use places high demands on working memory.
Headphone jack		
Sensors		
Accelerometer		

Table 6.3.: Constraints for mobile development [5]

6.10. Determine the Target User Group

Before starting with any design or development phase, it is crucial to know the target audience for a product. In general, these users have specific goals in mind, whether it is buying a new TV, or looking up the weather forecast for tomorrow, each product should offer a convenient possibility to fulfill this specific goal. The primary experience group should also be determined: is the app is designed to serve inexperienced to average users, or it is made for an expert? Most mobile app developers will focus on a broad range of user-groups in order to maximize their user-base.

To provide a good UX for inexperienced user an app should[72, 84]:

- Follow platform design guidelines
- Avoid too many navigation options on a single screen
- Provide an introduction during initial startup

Experienced users that want to effectively use your app should be served with:

- some kind of expert mode or extended functionality
- options for customization
- additional possibility of gesture navigation

In general users want to:

- Get right started with some action to get closer to their goal
- Have an obvious solution/explanation. If something is not obvious, they will not use it
- Get immediate feedback and fast results
- Consistently follow the same schemas for using your app

7. Usability and User Experience

"A product has no intrinsic usability, only a capability to be used in a particular context.[1]". A central statement to keep in mind about usability.

We use hundreds of products every single day. Some of them are very easy to use, while others turn out to be quite challenging or even frustrating. The user experience and respective usability is very important, and often a key factor for the success or failure of a product [5].

The usability, or user friendliness of a product, defines how well a user can interact with it. Every product has it. Cars, cell-phones, web sites, backpacks and even a ketchup bottles. The focus of this thesis is on the usability of software and especially on the usability of touch based apps. It also covers the term user experience (UX). UX includes some aspects of usability, but presents a more holistic view and also places focus on emotions and feelings of the user [69]. In industries the term UX often includes usability or sometimes is even used as synonym. In the eyes of researchers, UX is a very subjective and separate term. In any case, usability and UX are two terms that are strongly intertwined, each one is influencing the other [70, page 13]. The following sections will highlight the ways they are defined in industry standards. We will provide general design principles for user interfaces as well as tablet and smartphone specific usability and UX issues.

7.1. Definitions for Usability

Definitions for usability can be found a lot. The most import standards are defined by the International Organization for Standardization (ISO):

Ergonomics of Human System Interaction (ISO $9241-11$ as well as the ISO $9241-210$) defines usability as: "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use." [73][72][69].

In addition to the ISO $9241-11$ the Common Industry Specification for Usability-Requirements (CISU-R) [74] defines:

- The context of use
- Performance and satisfaction criteria
- The test method and context for testing

[1] http://www.usabilitynet.org/tools/r_international.htm#9126-1

A criticism according to the previous definition is that there is no influence of time and therefore does not consider the evolution of usability over time. Addressing that the ISO/IEC 9126 includes learnability which reflects a product's ability to be easily understood. Jakob Nielsen went event one step further and not only added learnability but also memorability into his usability framework. They should address time dependent usability attributes [8].

The ISO/IEC 25010 supersedes the ISO/IEC 9126 and builds a more complete model for software quality requirements and evaluation. Here the usability is covered in two different ways.

Usability - Product Quality Usability is part of the product quality. Besides this, the product quality consists of: Functionality, Efficiency, Compatibility, Reliability, Security, Maintainability and Portability. With respect to the scope of this work, we will only go into Usability containing:

- Appropriateness
- Recognizability
- Ease of use
- User error protection
- User interface aesthetics
- Technical learnability
- Technical accessibility

Usability in Use - Quality in Use The quality in use model covers usability aspects in a dynamic context. It is a contextual view on usability and covers *usability in use* extended by the *flexibility in use* and *safety in use* [75][73][15].

They can be further refined:
Usability in Use:

- Effectiveness in use
- Efficiency in use
- Satisfaction in use
  - Likability (satisfaction with pragmatic goals)
  - Pleasure (satisfaction with hedonic goals)
  - Comfort (physical satisfaction)
  - Trust (satisfaction with security)

Flexibility in Use:

- Context conformity in use
- Context extendibility in use
- Accessibility in use

Safety:

- Operator health and safety

- Public health and safety
- Environmental harm in use
- Commercial damage in use

Summarized usability is an objective view that defines mainly the achievement of pragmatic goals by using the product. Pleasure also brings additional subjective influences bound to the user‘s feelings while using the product [15]. This follows a direction of user experience focusing mainly on these aspects (see next section).

7.2. Definitions for User Experience

The UX is essentially a holistic view on the quality of the experience the user has. In contrast to usability, the term user experience is more complicated and abstract, with its main focus is on emotions and feelings [76]. Feelings when real users interaction with the product, but also sentiments they have in general about a product or a brand. This includes pleasure, surprise, satisfaction, stimulation, frustration, delightfulness and many more [70]. Hassenzahl et al. [77] summarizes the fulfilment of psychological needs as hedonic quality of a product. In addition, task-oriented qualities that serve more pragmatic functions remain. They are also covered by usability criteria. The influences on the experiences the users have are dependant upon many factors: The user interface, the performance of the software, the smoothness of animations, sound effects, consistency, etc. but can also be influenced by the users mood, his context (or environment), his previous experiences with products and so forth.

There is still no generally accepted definition for UX. The definitions we found vary mainly in their point of view on the topic. Some of these definitions are provided in the following section.

Microsoft[2] defines user experience on their msdn glossary as: *"An activity of encounter by a computer user with the auditory and visual presentation of a collection of computer programs. It is important to note that this includes only what the user perceives and not all that is presented."*

The definition by Lauralee Alben [78] president of Alben Design LLC and a well-known design consultant is: *"All the aspects of how people use an interactive product: the way it feels in their hands, how well they understand how it works, how they feel about it while they‘re using it, how well it serves their purposes, and how well it fits into the entire context in which they are using it."*

"A consequence of a user‘s internal state (predispositions, expectations, needs, motivation, mood, etc.), the characteristics of the designed system (e.g. complexity, purpose, usability, functionality, etc.), and the context (or the environment) within which the interaction occurs (e.g. organizational/social setting, meaningfulness of the activity, voluntariness of use, etc.)" is the definition for UX by Hassenzahl and Tractinsky [77].

The UX Book [79] *"User experience is the totality of the effect or effects felt by a user as a result of interaction with, and the usage context of, a system, device, or product, including*

[2] http://msdn.microsoft.com/en-us/library/bb246417.aspx

the influence of usability, usefulness, and emotional impact during interaction, and savouring the memory after interaction. Interaction with is broad and embraces seeing, touching, and thinking about the system or product, including admiring it and its presentation before any physical interaction."

One of the most important standards in this field is the ISO $9241 - 210 : 2010$ *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems.* It defines user experience as *"all aspects of the user's experience when interacting with the product, service, environment or facility"* [80] or *"a person's perceptions and responses that result from the use or anticipated use of a product, system or service"* [69].

But researchers [81] often think this definition is to abstract and therefore have created their own. In light of the fact that UX is, per se, a term describing subjective, dynamic and personal emotions there probably never will be a single, concrete definition for this term.

7.3. General Definitions

From Barnum et al. [72] general definitions that will be used throughought this thesis:

Usability Testing - the process of learning about users users by observing their usage of a product that accomplishes specific goals of interest to them.
Usability test - a single usability testing session.
Usability study - the total number of testing sessions.
User - the person who is using the product.
Product - a catch-all term to refer to any element or component of the design that contributes directly or indirectly to the user's experience. In our case the product will be a mobile app.
Interface - the part of the product that the user interacts with to accomplish tasks.
Specific users - not just any user, but the specific demographic for whom the product is designed.
Specified goals - these specific users must share goals for the product, meaning that the product's goals represent their goals.
A specific context of use - the product must be designed to work for the environment in which these users will use it.

7.4. Acceptability

Sometimes it might not be enough to make sure that a system offers a good UX respectively usability. It might be useable, but if it is socially or practically not acceptable to users they might not even bother using it. Jakob Nielsen iterates this problem in [8]. He uses the term *System Acceptability* as the overlying topic to keep in mind. The relationship to usability is visualized in Figure 7.1.

It is important to know about usability related topics like acceptability. Especially for new kind of apps, this might be a key point to be considered [82][83].

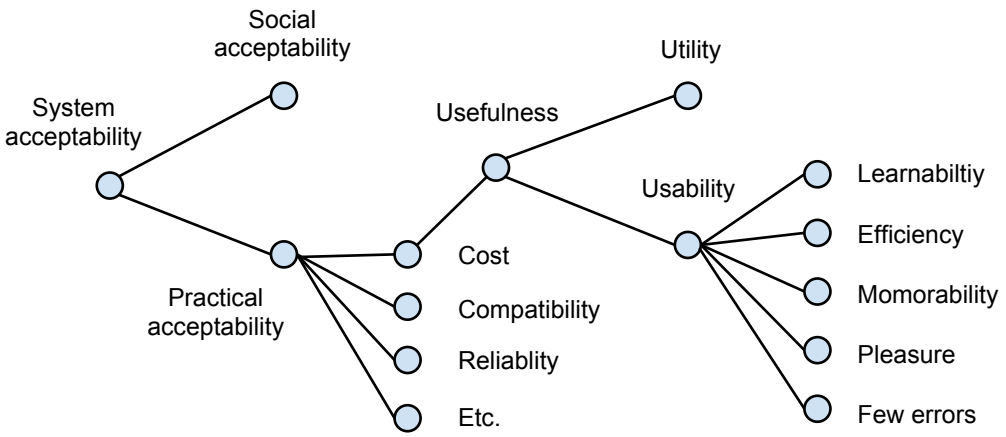

Figure 7.1.: System Acceptability by [8]

8. Usability and User Experience Evaluation

"Focus on the User". This statement is crucial. The user is always the center of interest. With the user comes the profits, and that is why it is important that they enjoy their experiences with a product. In order to find issues, the usability expert has to address the concerns of consumers, thus, usability testing is paramount. Observations and questions can reflect how users are doing when they interact with a product. Unfortunately, both user experience and usability are not directly measurable. But, thankfully, certain metrics can quantify some measurable indirect indicators (e.g. task completion time for efficiency). One of the most important phrases for every test user to hear before an evaluation should be "We are testing the product not the user!" [72, 10].

A basic categorization of usability evaluations contains two basic types [24, page 5]]: Usability Inspection and Usability Testing. This classification is based on how the product is tested. Although the basic principles for UX evaluations are very similar, they are not to be confused with one another. Section 8.5 contains a short introduction for some UX evaluation methods.

8.1. Usability Testing

Usability testing describes evaluation methods that involve users, who are interacting with the test object. This class can be further divided into two main types of testing methodologies:

1. Formative Testing
   Formative testing is normally conducted multiple times during development, with small groups of test users. The primary goal is identification of any major usability issues in order to repair them quickly. These tests are also useful in formulating better understanding of users and their requirements. Typically, only a small test group is needed [80][48, 5][72].

2. Summative Testing
   Summative testing determines quality metrics. This kind of evaluation is primarily used for products in nearing or at completion. It is also useful in comparisons of alternative designs and achieved usability requirements. The measurements must be reliable to ensure accurate results, and an appropriate sample size (test users) is important for the desired rate of confidence. The selection of tasks for the test should be reflected in real world usage examples [80][72].

   Furthermore, a formal and informal summative evaluation can be distinguished. Formal summative evaluations have empirical character to compare design hypothesis factors within a controlled environment. An informal evaluation can be conducted with a smaller sample size, and typically provides only summarized statistics [79].

Another distinction by Preece [70, page 437] distinguishes two types based on the environment where the test is conducted:

1. Usability Testing and Experiments
   A chosen sample of users perform tasks in a controlled environment. This method works well for finding usability issues in a predefined context. External influences are minimized by the testing moderator, who is in control of the whole process.

2. Field Studies or Field Testing
   A (virtually) random sample size test the product in a real-world environment (outside of the controlled variables and conditions of a laboratory). This provides good insight on how the product will be used as well as context. Some disadvantage exists with lack of control over the testing environment, and possible increases in costs.

[70, page 437]

In addition to categorization defined by the number of involved users, usability tests can also be classified by how they relate to their provided results.

1. Test that compare the results of different user groups against each other are called *between subject* test. Between-Subjects studies compare results between different subjects or control groups. For example, they can be used to find respective efficiencies between younger and older user groups, or the differences in satisfaction between them. To keep the variables across a group at a minimum, a larger sample size should be used. Between Subject studies are advantageous in that any carryover effects have no influence, because they impact all groups in the same way [84].

2. If each group of testers is consisting of the same users, the setup is called *within subject* [84]. Measurements of learnability, or comparison of multiple designs are examples for this type of study. Differences between groups are irrelevant, but possible carryover effects must be considered. They might occur due to fatigue users or unwanted practicing effects. To avoid this techniques like counterbalancing are required.

3. In addition to Within-Subjects, a Between-Subjects Study or combinations of the two can also be utilized. These studies contain at least one shared factor (e.g. age), and also a within-subject factor (three trials). This allows multiple questions to be addressed with a single study [6, 18-19].

8.1.1. Thinking Aloud

A very common strategy for small, formative studies is a so called "Thinking Aloud" test. This is a very powerful method for gauging how users feel while using a product. This method can be used to garner much more observable data. But the process (of thinking aloud) will also influence some other results, such as the time of task completion [72, 19].

8.1.2. Wizard of Oz Evaluation

This is a method of testing concepts for an application without the need for a functional prototype. The user interacts with a system operated by humans, giving them the impression of using a functional system, and is able to give any necessary feedback. One notable downside of this technique, is that it is not very useful for performance-centric or time-critical applications [85].

8.2. Usability Inspections

These inspections define a second class of usability evaluation methods. Instead of control users, experts investigate the system, product, or service, in order to find usability problems. It is usually a faster approach, but it tends to miss some predictable problems. The very widely-used heuristic evaluation and cognitive walkthroughs fall within this category of evaluations [70, page 506-514].

They have been described as unsuitable for mobile app evaluation because they fail to consider the context of mobile users (see Section 6.6).

Usability inspections can still be valid to some extent, due to the obvious fact that small-screen apps are still software products with graphical user interfaces. The heuristic Nielson [86] originally defined are:

- Visibility of system status
- Minimize the user's memory load
- Match between system and the real world
- Flexibility and efficiency of use
- User control and freedom
- Aesthetic and minimalist design
- Consistency and standards
- Help users recognize, diagnose, and recover from errors
- Error prevention
- Help and documentation

These heuristics also relate to the design principles [70, page 26] that have been described in Section 6.7. Nielson, the inventor of heuristic evaluation suggested an adopted version for specific categories of products. For touch based apps the user interface and design guidelines for iOS and Android might be a good place to start. But for the scope of this work, usability inspection methods are not relevant for the purpose of generating concrete usability metrics.

8.2.1. Planning a Usability Evaluation

This section explains how to design usability evaluations. The team must decide what kind of test method to use, which environment the evaluation will be conducted in, and how many control group participants are necessary in order reach the test goals.

Type of Method

To find out what type of usability test or inspection is suitable, several questions must be considered, the first being: In what stage is the development process?
During early development, a small usability test would be the best choice. This can and should be repeated multiple times in initial stages. This early feedback from a test group can start future development off in the right direction.

Is real life performance a factor to be determined? If so, a field study would provide the most useful and accurate feedback.

Do we want to compare different design options, or do we want to evaluate how different user groups are performing?
In this case, the best choice would a Within subjects study, rather than a Between Subjects method.

Do we need quantitative results, or merely a list of known issues that must be addressed in future updates?
Formative testing provides metrics and valuable quantitative data, therefore a larger number of test participants is necessary. For a simple list of usability findings, a summative study with a few participants is sufficient.

How much money do we have to fund the Study?
The available budget always plays a large role on a project. This often limits the number of test users that are involved and how much investigation can be conducted. Of course, most large studies are summative, and most small studies are formative, but with budget considerations, this might be adapted to suit any current needs. A complex application with many usecases and a very large and diverse userbase might need a formative study with a larger test group. Conversely, a very limited budget may result in a summative study with a smaller group of test users. The resulting metrics will not be as sound as values gathered from a large study [72, 15-23].

Type of Participants

This is dependant on the available budget and desired goal of a study. The selected user group should either represent a larger population or, better yet, the desired target audience. This is an essential requirement for further calculations and assumptions [6, 15-16].

Selection of Participants

There are several methods of conducting a usability study [6, 16-17].

Random Sampling
Everyone inside the target population is a potential choice, and is selected via a random number generator.

Systematic Sampling
Selecting participants by predefined criteria.

Stratified Sampling
Reflects a larger population for given characteristics, subsamples are created that represent these characteristics correctly.

Samples of Convenience
Essentially anyone can participate. These test users are selected out of convenience. They might be individuals already known by the testing team, or have already attended a previous study.

Divide up Groups of Participants

If certain groups of users should be compared, they should be divided by characteristics that might have an considerable influence. Possible categorizations are [6, 16-17]:

- Demographic characteristics (age, gender, ...)
- Level of domain expertise
- Level of knowledge in other relevant fields
- Frequency of use
- Education

Number of Participants

The number of participants plays a major role in influencing error tolerance of results. The number of needed participants depends largely upon the desired goal/type of the study. For formative studies, as previously mentioned, less participants are needed. Most literature suggests at least $3-5$ users to identify most usability problems in an early project stage [72].

Hartson [79] stated that 80% of UX issues can be found with only 4 to 5 participants. Such a small control group has been recently criticized by researchers [6], but many guides still suggest it. Since formative studies are typically done multiple times during development, it is crucial to keep them simple and quick (and therefore inexpensive). For an ideal method in an agile development process, we refer to RITE that is mentioned in [72, 113].

In addition to the study type, the complexity of the system also influences the number of participants that should be chosen. More complexity should mean more participants. The comparison of many different user groups is also a valid reason to conduct a larger study.

For summative studies it is a good practice to define the number of needed participants by defining a desired confidence interval [6, 17]. A t-test distribution can be used to calculate the needed number of participants for a given level of confidence [84]. Table 8.1 gives an impression about changing confidence levels with a growing number of participants. The example is calculated with a 95% confidence level [6].

Number Successful	Number of Participants	Lower Confidence	Upper Confidence
4	5	36%	98%
8	10	48%	95%
16	20	58%	95%
24	30	62%	91%
40	50	67%	89%
80	100	71%	86%

Table 8.1.: Confidence Intervals Change due to Sample Size (95% confidence level) [6].

Counterbalance the Order of Tasks

To reduce the impact of between-task learning effects, the order (of tasks) can be made variable between participants. The order can be randomized, or distinct orders can be defined beforehand to ensure even distribution [6, 19-20].

Create Task-based Scenarios

Tasks reflecting a realistic usage scenario are best suited for conveying real life usage of a product. A Task is defined as a specific goal that has to be fulfilled by the test users through out the study. This helps to produce more replicable and easier to analyse data [72, 19].

8.2.2. Measurements

Why is it important to measure usability and user experience? And what metrics can be used? As we have stressed in this thesis, the main goal of a usability study is improving the product. Knowledge of the user‘s expectations and needs is only one part of the equation. The best way to evaluate whether or not improvements are effective is to determine the current state of usability and compare it to previous versions.

The second possible goal of a usability study is a comparison. Metrics will either be used to compare product versions, or verify that certain benchmarks have been met. To ascertain whether or not these goals have been reached, certain measurements must be made during a usability test. Usability and flexibility in use are measured by effectiveness, efficiency and satisfaction.

Effectiveness
- Task goal completion and Partial goal achievement
- Completion Rate
- Number of Errors
- Number of Assists

Efficiency
- Task (completion) time

- Resources used
- Productivity (Completion Rate/Mean time on task)
- Relative user efficiency (How long a user needs compared to an expert).

Satisfaction
As defined in Chapter 7, satisfaction is defined by:
- Likability
- Pleasure
- Comfort
- Trust

Satisfaction can be determined on task-level as well as test-level. Questionnaires gather subjective measures to determine user satisfaction. Many of these questionnaires collect measures like usefulness and ease-of-use. Some of the most important standardized questionnaires are stated out in Section 8.4. Bevan [87] also points out that the achievement of pragmatic goals (Likability) and those of hedonic goals (Pleasure) can be determined with UX questionnaires that cover those topics (like AttrakDiff).

Learnability Measures: To measure learnability, a comparison of efficiency measures can be done in certain time intervals. The downside of this approach is that it can be very time consuming. Other ways to collect measures for learnability are questionnaires like the USE that include questions regarding this issue.

Expert-Based Method Measurements
Although this will not be covered in detail, we want to give a short insight to these measures. Expert-Based methods do not include that amount of samples to produce reliable metrics. Nevertheless numerous measures can give further details about the usability of a product interface. These measures include [80] [48, 5]:

- Number of violations of guidelines or heuristics
- Number of distinct problems
- Percentage of UI elements fulfilling specific guidelines
- Whether the interface fulfils requirements (e.g. Low Interaction Coefficient)

8.2.3. Types of Variables

Every usability test has two types of variables: Independent and dependent. All stats manipulated to answer specific research questions are called independent. This can be, for example, the age of users. The outcome and therefore the measured variables are called dependent. They must all be logically connected to the desired research goal [6, 20].

8.2.4. Calculations

Confidence Intervals

For confidence intervals the following components are important.

The confidence Level:
Describes how accurate the statistics are. The value in Percent defines how often the unknown mean value will be inside the confidence Interval if the experiment is repeatedly conducted [84]. For calculation, researchers usually choose a Confidence Level of 90 or 95%.

Variability:
An estimation calculated from the standard deviation.

Sample Size:
Defines the number of participants when conducting a usability test.

In addition to these values, it is necessary to provide an estimation value for the mean in order to calculate a confidence interval.

Using these data the confidence Interval can be calculated with[1]. For further details about formulas and calculation i refer to chapter 3 in [84].

Binary Success Data Confidence For binary success data (like Task-Completion rate) a confidence Interval can be calculated by using a binomial confidence interval [6]. Especially for small sample sizes Tullis refers in his book [6] to an adjusted version of the *Wald Method*. This is also suggested by Sauro [84] to get better results for small sample sizes. A web version of a confidence interval calculator is available[2]. In tests where more detailed data is needed it may be necessary to define multiple levels of success or failure.

Rating Scales Confidence For likert scale ratings that come into practice for usability questionnaires the best solution for calculating confidence intervals is utilizing t-distributions [84, 26]. The mean and standard deviation of the samples are calculated. The sample size is also accounted for.

$standard\_error = \frac{s}{\sqrt{n}}$

The t-critical value (for given $alpha = 1 - confidencelevel$ and $df = n - 1$) must be determined[3] and inserted into the formula:

$margin\_of\_error = tcritical\_value \cdot standard\_error$
The confidence interval then is twice the margin of error, thus the mean $\pm$ the margin of error.

[1] http://www.measuringusability.com/ci-calc.php
[2] http://www.measuringusability.com/wald.htm
[3] www.usablestats.com/calcs/tinv

Calculations Based on Time Data

For time measures (e.g. like Time-on-Task), there are typically a few calculations to be done: Average, median, geometric mean, and confidence intervals for each task [6]. Each of these values can give certain insights for the measured time values or minimize unwanted biasing. Any large differences in task completion times between users and therefore variability may warp the average. It makes sense to compare this value to the median or the geometric mean (suggested by Sauro [84]). The calculation of a confidence interval is done using the geometric mean estimation for the mean value.

Calculations Based on Errors

Errors are defined by the number of mistakes a user makes during the execution of a test. This may be anything from a click on a wrong navigation element, an average input to a form, or a failure to perform the correct action. An error rate (number of errors for a task/number of error opportunities) is often used for these occurrences. Weights might also be used according to the severity of an error [6].

Calculations for Efficiency Measures

Often-used measurments for efficiency are the number of actions used for completing a task. Another common value used to indicate efficiency is the ratio of the task completion rate to the mean time per task [6].

Calculations for Learnability

Learnability measures are done by comparing performance measures over time. Very often these performance measures are efficiency related (e.g. time-on-task, errors, number of steps, task success per minute).

8.3. Other Methods

8.3.1. Verbal Behaviours

These are simply things that users might say (intentionally or unintentionally) during a usability test. These expressions can be logged and categorized as useful data that can be used to make comparisons across design iterations [6].

8.3.2. Non-verbal Behaviours

Similar to verbal behaviour, the users may express non-verbal language such as frowning, smiling, fidgeting, leaning close to the screen, rubbing the head, etc. Similar to vocal expressions, they also can be logged and categorized. A comparison of their frequency across design iterations might bring additional insight to the data collected from verbal behaviour. But Tullis [6] points out that these measures only make sense for special kinds of applications.

More advanced methods to evaluate usability and/or user experience that may require additional equipment are[6]:
- Video-Based Systems
- Eye-Tracking
- Electromyogram Sensors
- Skin Conductance and Heart Rate

Considering factors of cost-efficiency, these methods will not be included in the scope of this thesis. For any further details we refer to the book "Measuring the User Experience"[6].

8.4. Usability Questionnaires

Questionnaires are a powerful tool to estimate subjective usability metrics (like satisfaction) [79]. They can be done before a usability test, after a completed task (post task questionnaire) and after the usability test. There are also some standardized usability questionnaires which have some further advantages:

- Standardized measurement for an objective comparison to other results
- Increased reliability and replicability
- Introduces the usage of statistics
- They are reusable by others so they do not have to create their own questionnaire
- Communication is made more effective
- Scientific generalization

[84, 186] Thus we would like to introduce some important post study as well post task questionnaires.

8.4.1. Post Study Questionnaires

Questionnaire for User Interaction Satisfaction (QUIS)
The QUIS is a licensed questionnaire with a global reliability of 0.94. In addition to 9 specific interface factors there are also sections containing demographic and an overall system satisfaction questions. There are two versions available, one containing 41 items and a larger questionnaire containing 122 items.

Software Usability Measurement Inventory (SUMI)
This 50 item questionnaire is also licensed, but has a free option for students. With them it is possible to determine the global scale based on 25 of those items and 5 subscales (Efficiency, Affect, Helpfulness, Control, Learnability). The general reliability is stated to be 0.92.

Post-study System Usability Questionnaire (PSSUQ)
Version 3 of this questionnaire contains 16 questions with seven scale ratings, resulting in 4 scores: An overall score and 3 sub scores (System Quality, Information Quality, Interface Quality). The PSSUQ can be used without any license and its general reliability is 0.94.

Software Usability Scale (SUS)
The easiest and widely-used post study usability questionnaire. It requires no license, and creates an over-score that ranges from 0 to 100, with 10 items containing form results in a 0.92 general reliability value. It uses 5 scale steps and alternating positive and negative tone questions. The SUS was developed by John Brooke in 1996 [88] [6] [84]. How can SUS scales be calculated? This is contained in Section 9.2.9 and 9.2.9.

Beside those Sauro[84, 213] also mentions additionally questionnaires for website usability evaluation:
WAMMI (Website Analysis and Measurement Inventory), SUPR-Q (Standardized Universal Percentile Rank Questionnaire) and a website usability evaluation framework by Wang et al. [89] that is based on the investigation of ease-of-navigation, speed, and interactivity to address the usability of websites.

8.4.2. Post Task Questionnaires

These questionnaires are conducted during the usability study or in lieu of a task. They focus on a lower level of the user-satisfaction than post- study questionnaires that summarize the overall experience. By analysing the results of single tasks, problems can be identified on a more detailed level [84, 213].

After Scenario Questionnaire (ASQ)
The ASQ is a freely available 3 items questionnaire. The questions are rated on a 7 point scale and they consist of[84, 213]:

1. Overall, I am satisfied with the ease of completing the tasks in this scenario.
2. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.
3. Overall, I am satisfied with the support information when completing the tasks.

Single Ease Question (SEQ)
As the name suggests, this is one question that is asked after a task is completed by the user. The single question is how hard it was to complete the task. Users can answer in a typically 5 or 7 point scale. Evaluations of the SEQ showed correlations with SUS scores, as well as with performance metrics[84, 214].

Subjective Mental Effort Question (SMEQ)
The SEMEQ has a rating scale from 0 to 150 with nine marked labels starting from "Not at

all hard to do" to "Tremendously hard to do". The marks start slightly above the 0 point scale and end at the 110 Points [84].

8.4.3. Pre Study Questionnaires

Vaananen-Vainio-Mattila et al. [90] points out the importance of pre study questionnaires, especially for UX analysis where the user‘s expectations should be considered. They can additionally be used to collect general data about the user (age, education, etc.).

8.5. User Experience Evaluation

We have highlighted usability metrics in Section 8.2.2. For user experience metrics, evaluation is more difficult. Experts still do not agree on whether measurements for emotional responses of users can be accurately determined at all. However, researchers have defined methods to determine estimates of the actual user experience. These ratings are often calculated by evaluating standardized questionnaires filled out by participants. Other methods involve the usage of visual elements, like different kinds of emoticons to capture the user‘s response. In this section we will introduce some important elements that are proven to be applicable in the industry.

8.5.1. Usefulness, Satisfaction, and Ease of Use Questionnaire (USE)

Arnold Lund developed a post-study questionnaire to quantify the user‘s impressions on the usefulness, ease of use, ease of learning, and the satisfaction in systems across many domains. He uses 30 questions and a 7-point Likert scale [6].

8.5.2. AttrakDiff

AttrakDiff is a questionnaire containing 28 questions that cover hedonic measures as well as pragmatic qualities. It was introduced by Hassenzahl et al.[91] and used to analyse the user experience of mp3 player skins. Hassenzahl affirmed that users not only want usable products, but also desire those that fulfil their needs for stimulation and identity, in reference to hedonic quality. AttrakDiff can be used to generate a measurement for some aspects of usability as well as the user experience of products. In more detail, it generates measurements in the following dimensions:

- Pragmatic Quality
  Usability or goal oriented quality
- Hedonic Quality - Stimulation (HQ-S)
  The quality of the product to stimulate the user by e.g. offering engaging functionalities, interesting content or new interaction and presentation methods.
- Hedonic Quality - Identity (HQ-I)
  The possibility of users to identify themselves with the product.

- Attractiveness (ATT) A subjective measure for the overall attractiveness of the product based on the users perception.

AttrakDiff is seen by its inventors as an instrument for research oriented evaluation techniques [91]. It has proven to generate promising results in several studies [27][80]. Hartson [79] also mentions that it can be ideally suited as an addition to traditional usability questionnaires like SUS or QUIS as explained in Section 8.4.1.

8.5.3. User Experience Questionnaire (UEQ)

Based on Hassenzahl's user experience Framework, an additional questionnaire for quantitative analysis of UX factors was developed by Laugwitz et al. [92]. The objectives for this questionnaire have been a quick assessment that is primarily designed to be used in addition to expert evaluations or usability testing. Thus the UEQ focuses more on subjective impressions and therefore user experience. Starting from an initial set of 221 adjectives formulated by expert proposals, this questionnaire was continually refined. In the end, 26 questions remained, covering measurements for Attractiveness, Perspicuity, Dependability, Efficiency, Novelty, and Stimulation. Like in many other questionnaires, a 7 point Likert scale is utilized. The reliability and validity has been shown in 2 Validation Studies [92].

8.5.4. PrEmo

PrEmo is a software instrument that enables to measure emotional responses in 14 dimensions. It is commercially licensed and designed to measure emotion as part of the UX quality [81]. This is realized by using graphically animated characters that show different emotions via facial expression and body language (similar to Emocards) [93].

8.5.5. Product Reaction Cards

Microsoft designed a method where user had to choose out of 118 cards containing words (adjectives). These positive and negative terms are chosen by the user to describe the interaction with the system after finishing the test [6]. Different versions might be conducted. For example users may only choose their top 5 cards or as many cards as they think are appropriate.

8.5.6. Others

Reaction Cards can be used to define a more fine grained description of the users experience with a product. In this set of 118 cards the user can pick, each card defines a phrase or an adjective [93].

Website Analysis and Measurement Inventory (WAMMI) is a tool for website evaluation [84, 223]. It is available as online service[4]. The questionnaire is based on the SUMI but focuses more on the evaluation of websites [6]. WAMMI is commercially licensed but brings a lot

[4] www.wammi.com

advantages like standardized benchmarks to hundreds of other websites, automatic report generation, analysis and interpretations. A similar commercially licensed online questionnaire is the Standardized Universal Percentile Rank Questionnaire[5] (SUPR-Q).

There are also other questionnaires that have not been investigated in detail because they require a commercial license. The Usability Metric for User Experience (UMUX) was kept apart from this work, because it covers only usability-related aspects and boasts a lower reliability than the SUS [84, 228].

[5] http://www.suprq.com/

9. Practical Part

The introductory chapters here that covered mobile ecosystems, and their different development strategies, outlined the complexity of mobile development. Based on that information, we started development for an e-Commerce application that could run on both Android and iOS tablets. This chapter (Section 9.1) covers the development process. The second part (Section 9.2) covers usability and user experience evaluation of this HTML5 based web-app. This provided very interesting insights and feedback. The main feature we addressed was a comparison of the tablet app and the desktop version, so a summative study with a mid-sized user base was utilized. Additionally, the evaluation provides metrics for comparisons to the usability of related products.

9.1. Development of a Tablet Optimized Web-App

This thesis was created in cooperation with NETCONOMY, a local development company offering multichannel e-Commerce solutions. Together with the smart devices team the development of an e-Commerce apps for tablet platforms was completed.

The main focus for our client was:

9.1.1. Project Goals

1. Targeting popular tablet platforms
2. Good user experience on all supported devices
3. Available as web app but also via app stores
4. Product range and functionality comparable to the desktop solution

9.1.2. Decisions and Development

In order to serve multiple platforms and remain available via a standard web url (project goal number 3) the decision had been made to use 100% state-of-the-art web technologies. Thus, the mobile application development framework, Sencha Touch, was chosen. It is based on HTML5 and development mainly utilizes JavaScript (see Section 6.3.1). Sencha Touch is preferred in contrast to jQuery mobile because it showed better performance characteristics and seemed to be better suited for developing a complex, single-page app.

Sencha Touch includes *Sencha Cmd* that provides many lifecycle management functionalities and therefore saves time during the development process. *Sencha Cmd* can be used for example

to scaffold new projects, minify the app or deploy it to a native build. Although we are additionally using Apache Cordova for native builds, this might be done by *Sencha Cmd* in the future. Sencha supports Android as well as iOS and therefore meets our project goal number 1. Additionally Blackberry and Kindle Fire devices (respectively all Webkit based browsers and WebViews) are supported. With respect to the current market situation the focus during development only relies on iOS and Android.

Additionally Sencha Touch [60] comes with the following features:

- Adaptive layouts
- Offline storage
- Extendable class system
- Smoother Scrolling and Animations
- Touch events and gestures
- Data Package (Models, Stores, Proxies)
- Large Userbase/Active Development
- MVC paradigm

Apache Cordova (PhoneGap) provided the native app packaging. Although Sencha has its own native packaging, Apache Cordova offers a wider range of flexibility, with an array of additional plug ins offered by their community. By wrapping the web app into a native app, users can find and download the app with Apple's App Store or Google's Play Store. All design templates for the appearance of the app have been created by an external company. Some designs caused problems, mentioned in Chapter 10.

The design principles and tablet usability constraints described in Chapter 6have been considered to provide the best possible user experience on a tablet device (Project goal number 2). In addition to usability, the code-quality was very important to our team. The best practices for today's software-engineering techniques have been used. They include:

Revision Control

A fundamental aspect of each development process is the revision control, or source control system. This guaranties that no completed work is lost and developers can move back and forth through revisions if necessary. For our development we used the mercurial open-source distributed source control tool. It offers a local repository as well as a remote master repository.

Code Reviews

Code reviews are very important. Non-trivial tasks have to be reviewed by other team-members in order to go into the development branch. Code reviews are very useful to increase the understanding of the framework for all team-members and additionally raise the overall code quality.

Agile Development

To leverage an agile software development lifecycle, the well-known scrum framework was used throughout the project. Although some aspects of scrum had been adopted to fit our needs. The core setup for our scrum process had been:
- 1 to 2 week sprints
- Daily scrum (standup) meetings
- Estimation meetings
- Sprint planning 1 and 2
- Sprint reviews and sprint retrospective

Many challenges are addressed during the development of software projects. This not only requires a well-planned project, but an actively engaged team. The next step for us was to determine how the current version of our tablet web app compared to the tradiotional web app. We wanted to evaluate the differences between the two frontends in respect to their usability and user experience measures.

9.2. Usability Study

The Smart Devices team at NETCONOMY had three primary motives for requesting a (summative) usability evaluation:

- The need for a qualitative comparison between the two implementations and how different they are performing on tablet devices.
- The results will define a quantitative measure to compare the current version with enhancements or future developments.
- Discovered issues can be useful for suggesting changes to the customer.

Sauro [84]explains that in industrial decision making processes a 95% confidence interval is very often more accurate than necessary, and can be adapted to 90% or even 80%. This is simply a trade-off between costs and an expected level of confidence.

In order to collect and compare usability and user experience measurements, an informal summative study should be conducted. Due to the limited time and budget, a data accuracy (confidence) of 90% was chosen to be sufficient.

Following Barnum et al.[72] we utilized the following test planning:

1. Establish test goals
2. Determine how to test the product
3. Agree on user subgroup(s)

4. Determine participant incentive
5. Draft the screener(s) for recruiting participants
6. Create scenarios based on tasks that match test goals
7. Determine quantitative and qualitative feedback methods
8. Set dates for testing and deliverables

9.2.1. Establish Test Goals

Analogue to the project goals for the app development, the first step before conducting a study is to define test goals [72]. In our case they have been decided to be:

- Define a baseline for measuring future improvements and for comparison with future developments of similar e-Commerce apps.
- Conduct a comparison to the standard web-version when used via a tablet device.
- Define a list of actions for future improvements.

9.2.2. How to Test the Product

In order to define a baseline and compare the two solutions we did a summative study. The web-shop as well as the new developed tablet optimized web-app are tested by each participant. This requires a larger number of test users depending on the desired reliability of the collected data. The number of participants necessary will be determined in the next section. The test sessions have been carried out as lab-based moderated sessions. This enabled us to track any usability issues that arose during the sessions. Another follow-up study with a changed mode of operation might be carried out in the future.

The following terms have been defined:
Independent variables: Number of Participants.

Dependent variables: Overall Usability Score, Overall UX Score, Pragmatic Goal Achievement, Hedonic Goal Achievement and Attractiveness Measures, Performance Measures (Time-On-Task, Success-Rates, Task Accuracy)

Hypothesis:

- The tablet optimized app will be (statistically) significantly more usable than the standard desktop website, when used on a tablet device.
- At least 90% of the participants will be able to complete all tasks.

The same users will test each of the competing apps (within-subjects design), which must be considered for statistical calculations.

In addition to to the post-study questionnaires, a one-question post-task usability questionnaires comes into play. This will give further insights into difficulties with specific tasks.

9.2.3. Agree on User Subgroup(s)

For this study a represented group of participants was chosen. Thus a distributed selection of users covering all of the following attributes was focused[6]:

- Expertise: novice, intermediate, and expert users
  In our case this attribute is split into experience levels concerning tablet devices and online shop usage.
- Frequency of use: frequent and infrequent users of e-Commerce sites
- Demographics: gender, age, education

Sampling of participants was done via systematic sampling, thus each participant was selected based on the previously defined criteria. The following statistics were considered in our choice of subgroups. Highest rates for online shoppers in Austria fall in age groups between 25 and 34 years. The exact numbers from the year 2012 collected by and can be found in the Appendix Section A.

Secondly the statistic offered by `statista.com` was taken into account which reveals that every second tablet user is in the age range of 25 to 34 (26%) or 35 to 44 (31%) [1].

Thus, we planned for the majority of all test participants to fall between these ages.

Statistics[2] about purchasing smartphone devices showed fairly even distribution between these demographics.

In Austria 52,4% of all male and 44,7% of all female[3] people have used online shops during the year 2012.

In order to get a comprehensive sample group our participants are chosen to reflect these known pre-conditions.

Determine the needed number of participants One of most important parameters of a usability test is the number of participants. For our within-subjects design the calculation has been done according to Sauro et al. [84]. Useful for estimating the mean of a normally distributed population are z-distributions. For small sample sizes the so called (students) t-distribution brings more appropriate results[84].

Assuming that the test statistic follows a t-distribution a a paired t-test can be used.

The formula for calculating the number of needed participants n is:
$n = \frac{t^2 \cdot s^2}{d^2}$
d ...observed difference
s^2 ...variance estimation of the sample
t ...critical value of t for the desired level of statistical confidence

[1] `http://de.statista.com/statistik/daten/studie/203080/umfrage/altersverteilung-der-ipad-nutzer-in-europa/`
[2] `http://www.thinkwithgoogle.com/mobileplanet/en/graph/?country=ch&category=MOBSHOP&topic=MOBSHOP_PURSPEV&stat=PURSPEV1&wave=wave2&age=a18_29&age=a30_49&age=a50&age=all&gender=male&gender=female&active=age`
[3] `http://www.statistik.at/web_de/statistiken/informationsgesellschaft/ikt-einsatz_in_haushalten/022212.html`

The desired level of confidence was 90%. Obviously this calculation needs an estimation value for the variance s (e.g. from previous test). In our case there is no such previous test session. Alternatively a fraction of the standard deviation can be used as estimation for the *critical difference (d)*. This is done by using an effect size e ("a standardized measure of the magnitude of an outcome" [84]). We used the suggested values of Cohen (1988): 0.2 small, 0.5 medium and 0.8 for a large effect. Where 0.5 (medium) was chosen to be reliable enough. The effect e can now be used to define d as a function of the standard distribution:
$d = e \cdot s = 0.5 \cdot s$
which results in the formula:
$n = \frac{t^2 \cdot s^2}{(e \cdot s)^2} = \frac{t^2}{0.25}$

Because t is dependent on the degree of freedom *df* which depends again on the number of users ($df = n - 1$) we cannot calculate it directly, but have to use iterations.

We start with z as approximation for t:
$n = \frac{z^2}{0.25}$

z is dependent on the desired confidence level (90%). $z = 1.645$(for a two-tailed test) using a z-score calculator[4].

The first iteration:
$n_1 = \frac{t^2}{e^2} = \frac{1.645^2}{0.5^2} = 10.8$
This rounded up to $n_1 = 11$
With the value for n and the desired level of confidence, the value t can be calculated for a two tailed test:
$df = (n - 1)$ ...the degree of freedom df is the estimation of n minus one.
$\alpha = 1 - c$ ...α is the two tailed probability value.
With $df = 10$ and $\alpha = 0.1$ a new estimation for t can be calculated with a t-score calculator[5].
$t_1 = f(df, \alpha) = 1.8124$ This t_1 can now be used to recalculate n with the original equation.

The second iteration:
$n_2 = \frac{(t_1)^2}{e^2} = \frac{(1.8124)^2}{0.5^2}$
Which results in $n_2 = 13.1 \rightarrow n_2 = 14$

The third iteration:
For $n = 14$: $df = 13$
$t_2 = f(df, \alpha) = 1.7709$
Which results in $n_3 = \frac{(1.7709)^2}{0.5^2} = 12.5 \rightarrow n_3 = 13$

The fourth iteration:
For $n = 13$: $df = 12$
$t_3 = f(df, \alpha) = 1.7823$

[4] z-score calculator: http://www.measuringusability.com/zcalcp.php
[5] t-score calculator: http://easycalculation.com/statistics/critical-t-test.php

Which results in $n_4 = \frac{(1.7823)^2}{0.5^2} = 12.7 \rightarrow n_4 = 13$ ... the iteration converges!

The estimation value for the number of needed test participants is 13 (with 90% confidence level). [84, 109-115]. This number is not large enough to generate the amount of reliable data that research guidelines would recommend. Nevertheless, when keeping confidence intervals in mind, this number is sufficient enough to provide information that is valuable for our previously defined goals[84, 10].

Additionally the requirement of *Sauro* et al. [94] is met who suggest a minimum of $10 - 12$ participants to provide reliable results with One-Question Post-Task Usability Questionnaires.

The Participants With the knowledge about our intended target population the following participants have been recruited. Table 9.1 lists their attributes.

Name	**Gender**	**Education**	**Age**	**Tablet Experience**	**Online Shop Experience**
Miriam H.	f	Kindergarden teacher	24	Low	Moderate
Clemens B.	m	Student-USW	26	Very Low	Very High
Anna H.	f	Student-Geography/History	20	Very High	Very High
Magdalena G.	f	Student-Musical Academy	20	Low	Low
Martin V.	m	Student-Telematics	27	Very High	Very High
Bernhard O.	m	Student-Biomedical Engineering	27	Very High	Very High
Rene G.	m	Student-Sports	23	Very Low	Low
Ulrike B.	f	Tax Officer	72	Very Low	Very Low
Bernd K.	m	Student-Telematics	26	Very High	Very High
Michael B.	m	Baker/Confectioner adept	54	Very Low	Very Low
Gerti B.	f	Baker/Confectioner adept	53	Very Low	Low
Stefan S.	m	Student-Telematics	26	Low	Very High
Florian S.	m	Student-Geography	27	Very High	Very High

Table 9.1.: Participants Attributes for the Evaluation.

The population contained 5 female and 8 male participants. Testing about 60% male participants seemed appropriate because of the app content (electronic retailer shop).

Distribution of participants among ages:
31% of the participants are between 16 to 24 years old, 4
46% of the participants are between 25 to 34 years, 6 participants
0% of the participants are between 35 to 44 years, 0 participants
15% of the participants are between 45 to 54 years, 2 participants

8% of the participants are over 55 years, 1 participant

Sub groups divided by experience levels:
38% or 5 out of 13 participants had a very high experience level in tablet and online shop usage.
38% of the participant had very low experience in both fields.
15% or 2 of 13 participants had low tablet experience, but high online shop experience levels.
7% or 1 of 13 participants had moderate experience in both fields.

9.2.4. Determine Participant Incentive

In our case the incentives provided to the participants had been:

- To be able to try out a not yet released product
- The chance to help improve this product to meet their expectations
- To get a small gift at the end of the session

9.2.5. Draft the Screener(s) for Recruiting Participants

For the study no screener was defined because we selectively choose people that were fitting into the recruiting requirements. So no invitations have been sent via email and therefore a screener was not necessary.

9.2.6. Create Scenarios based on Tasks that match Test Goals

We defined the following tasks for the evaluation based on the definition of test goals and the incentives of users. Furthermore they should reflect real life scenarios of an average user.

1) login/register (with predefined credentials)

2.1) Look for a specific Product (Mac Book Pro 15¨) and check the availability
2.2) Add this product to your cart
2.2) By a specific product (checkout process)
2.3) Check your order in your account page

3.1) Search for a product (Canon IXUS 105) and check the price
3.2) Reserve this product at a specific shop

4.1) Search which stores are near you
4.2) Check out opening hours of the nearest shop
4.2) Check out the route by car to the nearest store

5.1) Look for current offers on the site
5.2) Put an interesting product on your wishlist

To minimize carryover effects, we randomized the order of tasks (except task 1) as well as the order of apps tested.

9.2.7. Determine Quantitative and Qualitative Feedback Methods

For the determination of an overall usability measure a post-study questionnaire was chosen. The Simple Usability Scale (SUS) was our first choice, because it is widely used for valuable data in a variety of fields [95]. After reviewing hundreds of studies, a curved grading scale for mean SUS scores has been proposed [84]. Sauro and Lewis also recommended SUS (John Brooke [88]) as a post-study questionnaire. Research had shown its accuracy and quick applicability. A slight adaptation of the questions replaced the word "system" with "website" because some participants might not be familiar with the term web app (see Table C.1). The questionnaire is contained in Section C of the appendix. On a 5 point scale (additionally marked with numbers from 1 to 5) the user could choose from *Strongly Disagree* to *Strongly Agree* (*Stimme gar nicht zu* to *Stimme voll zu* in German). The calculation of a SUS score is explained in the following section.

To be able to differ the difficulties for each single task the Simple Ease Question (SEQ) post-task questionnaire was conducted (see Table 9.2).

Overall how hard was this task?

Table 9.2.: Single Ease Question (SEQ) after each task

A seven step scale reaching from *very hard not hard at all*(*Sehr schwer* to *Gar nicht schwer* in German).

Because we wanted to compare the UX between the two implementations, an additional questionnaire was also utilized: the promising AttrakDiff post-study questionnaire. It shows deep insight into to the overall view of the product. In contrast to the SUS, it was designed to generate UX based measures for pragmatic and hedonic qualities, as well as attractiveness. Table B.1 in the appendix shows how these questions are defined.

During the lab-based moderated testing, it was also possible to make note of any usability issues during the session. The frequency of these problems has been collected to provide a rating for the most distracting issues. Measures taken during these sessions will be covered in the following section.

Measurements Taken During the Evaluation

The chosen measurements reveal metrics for the effectiveness and efficiency of the apps. In reference to Section 8.2.2, the following measures have been taken for each task and participant:

- Task completion time,
- Task completion rate,
- number of errors,
- quantity of advices from the moderator

Measures for Safety will not be collected, because any risks to the user's health have been concluded to be irrelevant for this e-Commerce application. The only possible user risk worth noting would be during the payment procedure, but this is processed via an external service (Datatrans[6]) available inside an iFrame.

9.2.8. Set Dates for Testing and Deliverables

The dates for all testing sessions have been set on 19th January 2013 - 22. January 2013. The calculation of statistics and analysis of data will be completed on the 24th of January 2013.

9.2.9. Results

The following section covers the facts on resulting data that was collected during the study. This includes the results of questionnaires and basic calculations such as determining confidence intervals. What we learned from this information will be discussed in Chapter 10.

Results of the SUS-Questionnaire

Results for the System Usability Scale (SUS) questionnaire are stated in Table 9.3. The points for a single SUS questionnaire can be calculated following to [88]. For odd numbered questions (positive tone) the points are calculated as scale position minus 1. Thus, the points for an even numbered questions (negative tone) result in 5 minus the scale position. By adding up the single points for each question and multiplying them with 2.5 results in the SUS score.

For the resulting mean SUS Score a confidence interval can be calculated according to the description in Section 8.2.4.

Confidence interval for the mean SUS of the Desktop Version:
$standard\_error = \frac{s}{\sqrt{n}} = \frac{15.17}{\sqrt{13}} = 4.21$
The t-critical value(for alpha 0.05 and df = 12)[7]: 2.18
$margin\_of\_error = t - critical\_value \cdot Standard\_error = 2.18 \cdot 4.21 = 9.17$
So the confidence interval (for a confidence level of 95%) ranges from:
75.45 to 93.79

Confidence interval for the mean SUS of the Tablet Version:
$standard\_error = \frac{18.30}{\sqrt{13}} = 5.03$
The t-critical value is also 2.18

[6] http://www.datatrans.ch
[7] www.usablestats.com/calcs/tinv.

User ID	SUS Score Desktop	SUS Score Tablet
1	72.5	82.5
2	100	100
3	92.5	90
4	97.5	90
5	87.5	72.5
6	97.5	100
7	82.5	95
8	52.5	40
9	97.5	97.5
10	65	67.5
11	70	82.5
12	87.5	57.5
13	97.5	95
Mean	84.62	82.31
Standard Deviation	15.17	18.30
Min	52.5	40
Max	100	100

Table 9.3.: Simple Usability Scale (SUS) Scores

$margin\_of\_error = 2.18 \cdot 5.03 = 10.96$
In summary, the confidence interval (for a confidence level of 95%) ranges from: 71.35 to 93.26.

Results of the SEQ-Questionnaire

Results of the Simple Ease Question (SEQ) questionnaire gives additional insight on which tasks were the most challenging for our participants. A visualization is shown in Figure 9.1. The accounted difficulty for each single task is the mean overall ratings by test participants. As we pointed out in Section 8.4.2 the 7 point scale for the SEQ ranges from 1 (Very hard) to 7 (Very Easy).

The means for each general task (see Section 9.2.6) are shown in Table 9.4 for the desktop version and Table 9.5 for the tablet version.

Task#	mean	geometric mean	standard deviation
1	6.3	6.1	1.25
2	6.4	6.3	0.81
3	5.6	5.2	1.74
4	6.3	6.1	1.03
5	6.7	6.7	0.67

Table 9.4.: Desktop Version - Task Statistics for the SEQ

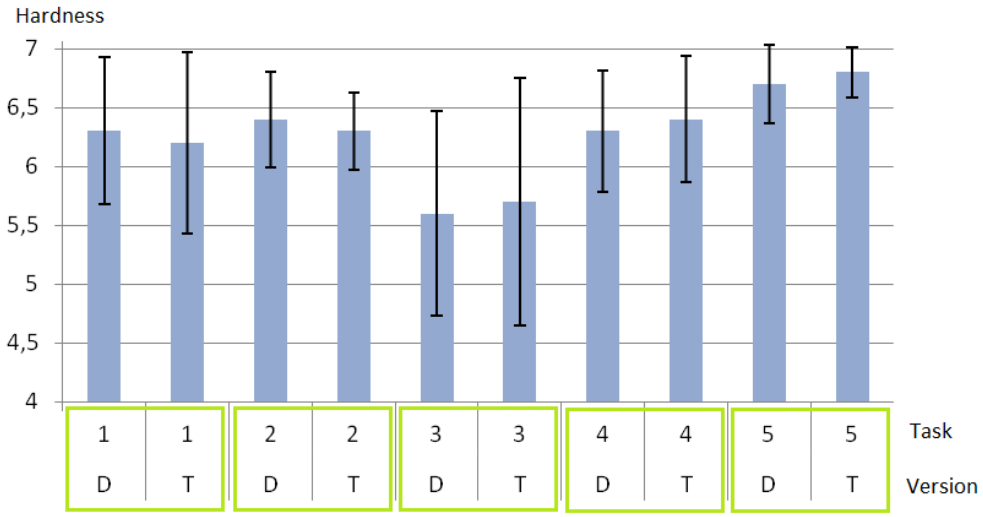

Figure 9.1.: SEQ Results: Mean Values and Standard Deviation for each Task (1-5) and App Version (Desktop D, Tablet T). Hardness varied from 1 (Very Hard) to 7 (Very Easy).

Task#	mean	geometric mean	standard deviation
1	6.2	5.9	1.55
2	6.3	6.3	0.65
3	5.7	5.0	2.11
4	6.4	6.3	1.07
5	6.8	6.8	0.42

Table 9.5.: Tablet Version - Task Statistics for the SEQ

When those two tables are compared the following conclusions can be made: The differences between the two versions are only small. Task one needed a lot of typing. Thus the difficulties with that task depend a lot on the skill set of the participants.

The lowest mean values for each version shows task # 3. this task was the most difficult one. But at the same time the high standard deviation indicated the highest differences between participants– thus, for some, the task was quite challenging while other participants experienced few problems. The reason for this are usability issues #6 and #7 which resulted confusion. The same problems also occurred in task # 4 (who has also a quite high standard deviation) but here the issues were already known for most participants.

Results of the AttrakDiff-Questionnaire

As explained in Section 8.5.2 AttrakDiff offers a finer grained scoring. The scores are divided into 4 main dimensions: Pragmatic Quality (PQ), Hedonic Quality - Stimulation (HQ-S), Hedonic Quality - Identity (HQ-I) and Attractiveness (ATT).

Figure 9.2 visualizes the summarized results. Shown values are mean values over all participants. Single scores are calculated by transforming the chosen value of the 7-point likert scale into

values from 3 to −3, with respect to correct mapping of attributes to dimensions. A value of zero defines an ordinary or average scale for this dimension. A value of zero basically defines an ordinary or average scale for this dimension.

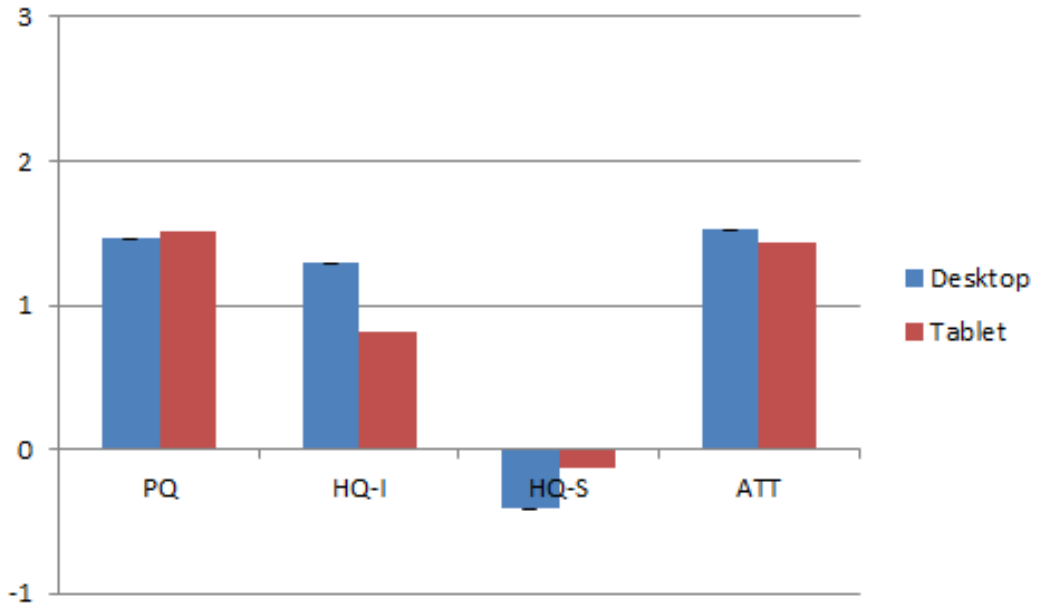

Figure 9.2.: AttrakDiff Mean Values. Projektteil A: Desktop and Projektteil B: Tablet Version.

PQ is the dimension that is mostly comparable to the usability of the product. Both versions scored around +1.5 points. On the one hand this result indicates a good rating to support the user in achieving his goals. But on the other hand the very high result of the SUS score (averages scores of 84.62 and 82.31) would anticipate a PQ score around 2.5.

The Hedonic Quality - Identity dimension showed a large difference between the tablet app that scored 0.81 and the desktop app with 1.29. This value indicates how indicates how strongly users identify with the product. A higher value for this dimension is important to tie customers to a product. This is the case if the product design, style and reputation match their values. Thus you need to know a lot about your target user group and their expectations. Here the tablet implementation show needs for some future improvements and during the sessions here often a better integration of social-web elements was mentioned. Also more personalized content and appearances would also bring vast improvements to this dimension.

The lowest rating was encountered for HQ-S, where also the large difference to the HQ-I value is notable. Resulting scores of −0.41 (Desktop) and −0.13 (Tablet) showed the highest potential for improvement. This can be done by motivating the user to keep his or her attention and interest on a high level.

The ATT dimension or attractiveness of the product reached a score above average and with 1.52 (Desktop) and 1.43 (Tablet) these results are nearly identical. A high value of attractiveness indicates a pleasant interface, a likeable appearance and a good overall impression. The focus of improvement for this dimension (when analyzing the word pair-scores) will be the attractiveness and appealingness of the product.

Results of Measurements

Following selected guidelines from [96] measures for effectiveness and efficiency are presented in this section. Satisfaction measures are covered by our questionnaires. Our measures for effectiveness include: Completion Rate, Errors and Assists. Efficiency was determined by: Task Time and Completion Rate/Mean Time -On Task.

Resulting mean-times-on-tasks gave also a very similar result for the tablet and desktop version. Although the standard deviation here is very high, in fact about 50% of the mean value. Figure 9.3 shows the mean task-times. The actual numbers are found in Table F.1.

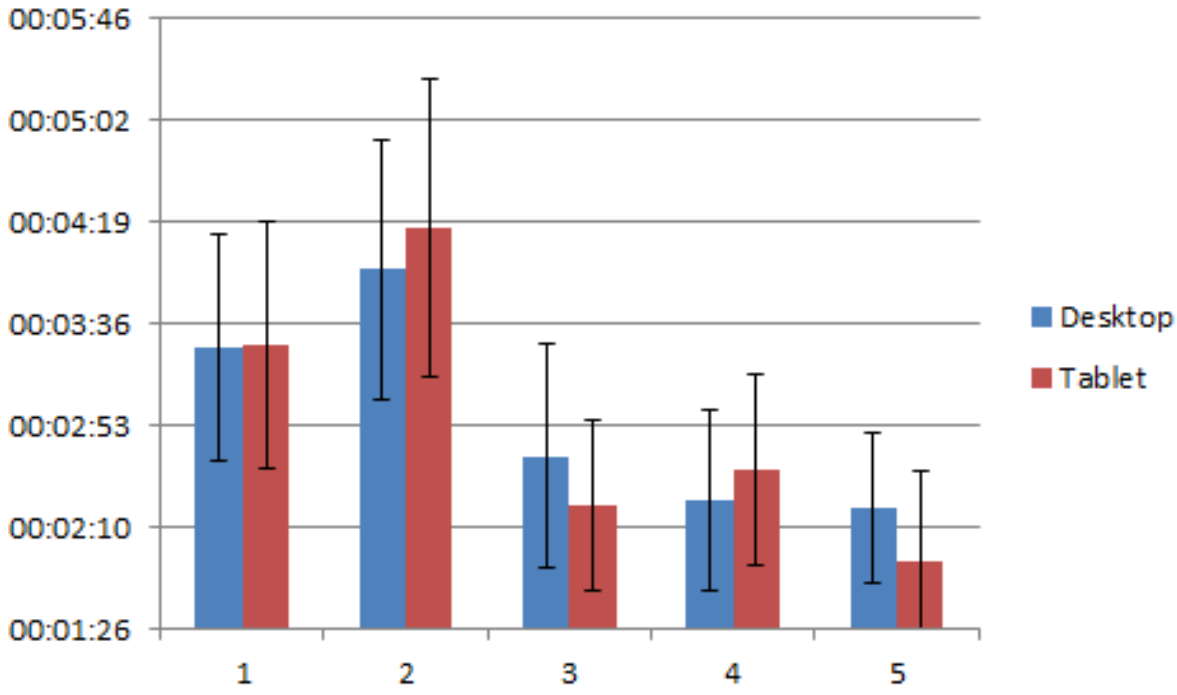

Figure 9.3.: Mean time on task for Desktop and Tablet Version.

Error-Rates and Completion-Rates are important measures for effectiveness. Their values are contained in Table 9.7 for the tablet optimized website and Table 9.6 for the desktop optimized site. Error Task relate to the number of tasks in percent that contained one or more errors.

Task #	Completion Rate	Unassisted	Assisted	Error Tasks
1	100%	84.62%	15.38%	61.54%
2	100%	84.62%	15.38%	61.54%
3	82%	54.55%	27.27%	45.45%
4	100%	81.82%	18.18%	54.55%
5	100%	81.82%	18.18%	72.73%

Table 9.6.: Completion and Error Rates - Desktop Version

Task #	Completion Rate	Unassisted	Assisted	Error Tasks
1	100%	76.92%	23.08%	76.92%
2	100%	84.62%	15.38%	84.62%
3	91%	63.64%	27.27%	81.82%
4	100%	63.64%	36.36%	90.91%
5	91%	81.82%	9.09%	45.45%

Table 9.7.: Completion and Error Rates - Tablet Version

The most important rates for our purpose have been the unassisted task completion rates. For those results the confidence intervals have been calculated to show the numbers in the context of their accuracy (Table 9.8).

Completion Rate	Low	High	Margin of Error	Best Estimate
100% (13 of 13)	0.7974	1.000	0.1164	0.9333
84.62% (11 of 13)	0.5654	0.969	0.2018	0.8
81.82% (9 of 11)	0.5115	0.9601	0.2243	0.7692
76.92% (10 of 13)	0.4906	0.925	0.2172	0.7333
63.64% (7 of 11)	0.3923	0.8967	0.2522	0.6667
54.55% (6 of 11)	0.2799	0.7875	0.2538	0.5385

Table 9.8.: Adjusted Wald Confidences for specific completion rates.

Determine Statistical Differences between Alternatives

A major question for summative comparison studies is: "Are the differences between alternatives statistically significant"?

The formula used for determining the significance for differences in a paired-t test presented in [84, 64] is:
$t = \frac{\hat{D}}{\frac{s_D}{\sqrt{n}}}$
$\hat{D}$ is the mean of the difference scores
s_D is the standard deviation of the difference scores
n is the total number of pairs of users
t is the test statistic

t can be used in combination with a t-table (were the degree of freedom $df = n - 1$) to get a probability for differing scores. A calculator is available online[8]

Differences of SUS-Mean values:
$\hat{D} = 8.46$
$s_D = 8.39$
$n = 13$

[8] Percentiles from the t-Distribution Calculator `http://www.usablestats.com/calcs/tdist`

$t = \frac{\hat{D}}{\frac{s_D}{\sqrt{n}}} = \frac{8.46}{\frac{8.39}{\sqrt{13}}} = 3.64$

Inserting this value into the t-Distribution Calculator (`http://www.usablestats.com/calcs/tdist`) with a Degree of Freedom (df = 12) the probability value for a two sided t-distribution: 0.0034 Which means that the chance that the two means are equal is 0.34%

Confidence Interval Around the Difference

Confidence intervals can help to identify how precise a result value is. For calculation researchers usually choose a Confidence Level of 90 or 95%. This may also be used to determine if any given difference is notable by the user [84, 66].
The formula for calculation is:

$\overline{D} \pm t_a \frac{s_D}{\sqrt{n}}$
$\overline{D}$ is the mean of the different scores
t_a critical value
n is the total number of pairs of users
s_D is the standard deviation of the difference scores

t_a can be calculated with a t-distribution and inserting a chosen confidence as well as the degree of freedom (number of users minus 1) see [9]
[84, 66][72]

For our SUS differences these values are: $\overline{D} = 8.39; t_a = 2.17881; n = 13; s_D = 8.39$ $\overline{D} \pm t_a \frac{s_D}{\sqrt{n}} = 8.39 \pm 2.18 \frac{15.17}{\sqrt{13}} = 8.39 \pm 5.07 = 13.46 and 3.32$
So we can be 95% sure that the real mean of difference is between 13.46 and 3.32.

Correlations Between Data

Neumann et al. [27] determined a positive correlation between the pragmatic scale of AttrakDiff and the mean task duration time.

Although this was also the case in our study, the correlations for this relation turned out to be fairly insignificant:
Desktop: $0,0315$
Tablet: $0,2456$

The correlation between mean task duration time per user for each of the two systems was: $0,8881$

Other interesting correlations are shown in Table 9.9:

[9] http://www.usablestats.com/calcs/tdist

Attribute	**AttrakDiff Dimension**	**Corr. Coeff. Tablet**	**Corr. Coeff. Desktop**
Total SUS	PQ	0,7868	0,7659
Total SUS	Total AttrakDiff	0,4581	0,4145
Gender	PQ	0,7428	0,6613
Gender	HQ-I	0,6369	0,8279
Gender	HQ-S	0,4680	0,6095
Gender	ATT	0,9288	0,7639

Table 9.9.: Correlations Coefficient findings.

A calculation of correlation-coefficients between SEQ scores and mean task-times did also not show a dependency.
The results for the Desktop version correlated with 0.0345
The results for the Tablet version correlated with $-0,1518$

A very high correlation could be determined between the mean SUS ratings and the standard deviation of SUS ratings:
The results for the Desktop version correlated with $-0,9566$
The results for the Tablet version correlated with $-0,9116$

Worth noting is the correlation between the SUS-score differences and the participants‘ preferred app. Here only differences came into account, thus values of participants that rated both apps with the same SUS-score have been sorted out. The positive or negative difference of SUS-scores has been correlated to the chosen app indicated by -1 for the Tablet version and +1 for the Desktop version.
The resulting correlation coefficient is:
0.5646
So only every second participant chose the version with the higher SUS score as his or her preferred app.

9.2.10. Formative Findings

A list of all formative findings as well as a summary of their frequencies can be found in Table E.1 and Table D.1 in the appendix of this work.

They conclude usability issues and bugs that have been found during the study. Although the focus has been on summative results, the additional information brings further possibilities for enhancement. Thus, for many of these problems a future release will bring a solution that addresses them.

Solutions for the most important ones for the Tablet version regarding their frequency of occurrence and interference of the user (severity).

- 9, unrecognized taps
  The problem lies in the Sencha Touch framework, because this exists mostly in the

Tablet version. The repair of this issue also needs to be addressed inside the frameworks touch handler.

- 7, Storesearch works only for zip codes
  The best solution would be to allow the search for arbitrary entries (zip, city, address).
- Small input elements
  Small input elements like radio buttons during registration should be made bigger and therefore easier to use with touch.
- Route/Fahrplan
  The terms "Route" and "Fahrplan" turned out to be misleading. The solution we made was to simply change the term "Fahrplan" to "SBB Fahrplan" and to swap the position inside the view. (Route on top because it will be used more often).
- The localization button
  Many users tried to submit their entered zip code using this button. The position of the button should be changed to be more apart of the input field.
- The content of the "Themenbühne" not touchable
  This is a CMS problem and needs to be done by the shop owner the right way.

Solutions for severe issues:

- 54 Submission of Order is confusing
  Change the final submission window, so the Checkboxes for the AGB/Newsletter and the Submit Button are on the final screen and not under the submenu.
- 53 Layout for Errormessage
  A redesign for the Error Popup window.
- Too much to write 51, 32, 33
  Prefill data for first name and last name (at least for the first address).
  Additionally the city can be chosen due to an entered zip code.
- Konto button for registration 50
  Rename 'Konto' to 'Registrierung/Login' if user is not logged in.

10. Lessons Learned and Conclusion

The results shown by the evaluation have been quite suprising. But with respect to the current status of the project and the compared web site the usability (SUS) score and user experience levels by the AttrakDiff questionnaire show that the app already has a good baseline.

Of course we expected the evaluation to show our tablet optimized app to be the clear winner on these kind of devices. But obvoiusly much less effort and budget was put into the development. Additionally this was the first mobile app project our department has developed. Thus a lot of new knowledge and experiences were collected during the last weeks. The potential for further improvement and enhancement is for sure quite big. And now that we have a current baseline for usability and UX measures we are also able to measure influences of future changes.

Another lessen we saw is that there seem to be two usergroups with quite different expectations and preferences. While one group is in general more comftable with a browser-based e-Commerce experience, the other group already prefers the advantages of a more app like experience. Our evaluation showed that these groups are currently about the same size. This might of course be changing in the near future. Mobile commerce business solutions are still only a side channel but their importance is indicated to be growing. Touch devices are going to be used more and more. Surfing the web and using online services are some of the most frequent tasks users perform [23]. Many companies react to this development and either try to optimize their website for these new devices or even provide a native app for their users. Also for e-Commerce businesses this is a very important consideration, although it is a significant additional investment. We also pointed out the importance in providing the best possible user experience to maximize the revenues through this new channel. A summative usability/user experience evaluation is necessary to provide valuable measures for these important product attributes.

Tablet and mobile friendliness websites have bigger touch areas. Thus, whole table cells can be touched and not only radio buttons or labels [48, 134].

10.1. Lessons Learned During the Development

This section covers which solutions to problems or improvements we would make for future developments.

10.1.1. Dependencies on Other Team

Because we were using the same backend as the desktop web-shop, there have been dependencies on the web development team. But in fact this is only one possible kind of dependency that is possible between teams. In our case this resulted in a lot of problems.

Therefore development teams that show such a deep interconnection and dependency should work together as closely as possible. There is absolutely no room for competitive thinking. If possible the teams should even be acting as one big development team. Times and deadlines should be in sync and sprint cycles as well as deployment dates should be at the same time.

10.1.2. Spatial Distribution of Team Members

Teams that do not work locally together face significant difficulties. Our team was located on multiple locations, with Vienna and Graz as locations for developers. These mainly are problems due to a lack of communication and information distribution.

In order to minimize these problems a guideline was introduced that enforced a rule that every action or change of state that could influence others has to be documented. Additionally the documentation has to have some kind of systematic approach, or it will be extremely difficult to properly locate information. Thus documentation took place inside our planning tool if it was related to a certain task. Therefore this project management tool should be something like an evolving version of the functional specifications. If it is very low level and only relevant for development purposes the information can be documented as a comment inside the code itself. Additionally should all necessary steps for certain tasks (like building the app, deployment, ...) be documented somewhere. This circumvents that only a single person is able to fulfil some task.

We used internet telephony services (Skype in our case) for the daily "standup-meetings". Also a very encouraging approach is to use video chats like Google hangout during the whole day, to create the impression of closeness to the other members. This also shortens communication times dramatically.

10.1.3. Testing

Automated testing is very important and should be done more frequently to minimize the repetitive task of manual testing. This is very time consuming, especially if a project aims to support many platforms. Even on our very small selection of operating systems and browsers each test has to be done on a minimum of 5 environments.
3 Browsers (mobile Safari, Android stock Browser, Chrome for Android).
2 Native Builds (Android, iOS).

For upcoming developments the following can be done:
- Ranorex Smoke Tests will be considered
- A promising testing tool is "Finding layout bugs" for automatic detection of possible problems

with the layout

Remaining problems are:
- Unit Tests for UI tasks mainly not possible
- Android brings in an huge device fragmentation and many OS versions.

10.1.4. Sencha Touch related Challenges

Custom Components

Our screen design was very unique and not designed to be implemented via adopted Sencha Touch components. Thus templates were implemented as custom components which resulted a lot of problems like the caching of components and retrieval via `Ext.getCmp()` did not work reliable any more. Our workaround for that was to use
e.g. `Ext.get(document.querySelector('.ordercontainer'))`
or `document.getElementById('RmaOverlayClose')` instead. The better and more promising solution for future projects would be to request desings that can be implemented via Sencha Touch standard components.

10.1.5. Platform Specific Problems

External Links on iOS

There was a problem with opening external links on the native built on iOS. This could be overcome by adding some helper code inside the PhoneGap framework. This helper intercepts links with match special patterns and routes them to the safari app[1].

Performance Issues on Android

We were facing problems with the old Android stock browser (pre ICS). As described in Section 4.3 the performance on the deprecated but still heavily used android stock browser is very bad. Thus we had to build in some optimization code in order to minimize this issue. Our solution was to minimize the size of the DOM. Thus all background structure was removed when overlays are displayed.

Orientation Problem on Android

The old Android browser has known issues with orientation changes[2]. This issue could be overcome by using a delay time before the dimensions have been detected again.

[1] https://gist.github.com/827979
[2] https://github.com/iosscripts/iosSlider/issues/117

10.1.6. Other Challenges

Like other software projects we were also facing some typical problems like: Grown Code resulting in very large controllers, confusing code constructs or violations to a strict MVC architecture. Here some refactoring sessions and code reviews must be executed to ensure future maintainability and extendibility of the code. Classic layout problems with localizations, especially with French translations happened a lot.

10.2. Lessons learned during the UX study

The results of this study gave interesting insights in users expectations and compared the two implementations for this e-Commerce application. Of course, due to the limited number of sample users the confidence interval for the results are pretty high. But for our purposes and the current phase of development this was sufficient enough to show up estimates for the current product qualities.

10.2.1. Test Goals

First of all we are going to examine if our previous test goals have been met.

The first one: "Define a baseline for measuring future improvements and comparison with future developments of similar e-Commerce apps.". In order to do so the calculated SUS and AttrakDiff scores provide useful measures for future comparisons.

Goal number two: "Conduct a comparison to the standard web-version when used via a tablet device." including observed differences will be explained in the Sections 10.2.2, 10.2.3, and 10.2.4.

Goal number three, "Define a list of actions for future improvements." will be covered in the Future Work section.

10.2.2. SUS Result Interpretations

Very unexpectedly, the tablet-optimized web app did not gain a statistically significant higher usability score. Respecting the comments of participants, the main reason was the more familiar structure and appearance of the desktop version. Our conclusion is that many users are simply not comfortable with online shopping via an app. Additionally there have been a lot of unrecognized touch interactions for the Sencha Touch app that annoyed some of the participants.

As shown in Table 9.3 the mean SUS score for the Tablet app was 82.3 against the 84.6 of the desktop version. Nevertheless 8 out of 13 participants preferred to use the tablet optimized app. The error margin was calculated with 0.2350 for 95% confidence (respectively 0.1639 margin of error for 80% confidence). Therefore the results show no statistically significant difference. This result is very surprising considering the significant differences in the appearance and behaviour of the apps.

10.2.3. Differences SUS to AttrakDiff

Contrary to the expectations that the AttrakDiffs Pragmatic Quality (PQ) score would behave similar to the SUS scores, the two measures turned out to have quite different magnitudes. While the SUS scored in around 85% of the best possible score the PQ score only reached 75%. A reason for this might be the different strategies for the determination of those questionnaires.

A calculated correlation coefficient between these two values scores showed a correlation of 0.7868 (Tablet) respectively 0.7659 (Desktop). Although this was expected to be higher, this showed a quite strong dependency between those values. The lower correlation values of 0.4581 (Tablet) and 0.4145 (Desktop) between SUS and the total AttrakDiff scores demonstrates that user experience measures are covering a more holistic impression on users than only goal oriented usability measures do.

10.2.4. General

The fact that the tablet-optimized app did not gain a statistically higher rating in our study posed the question of why such an event would occur.

The behaviour of participants and their comments gave the following insights: The Sencha Touch app felt more like a native app for users. The design of this version was reported to be much cleaner and simpler. But this was not seen as a positive aspect for all users. Some users implicitly expected less functionality provided by an app in contrast to a website. This was in contrast to others that appreciated the clean design and enjoyed the navigation through the app. This observation showed that there are people who prefer to use websites and others who prefer to install and use a mobile app.

Our conclusion is that it is very important for an e-Commerce app to provide the same features that are available on the web version. Therefore a smart design is needed in order to avoid overloaded screens. The gained insights also indicated potential for numerous improvements. Discovering where users have difficulties is the best starting point for finding what to improve. This is the first version of a touch based/tablet optimized implementation that has been developed for a customer. Considering this, the outcome is very promising. A positive summary can be given for the total outcome of the study. A good overall UX/usability for both apps (desktop Web-app and tablet Web-app) could be determined. This allows a high value outcome for both products.

10.3. Future Work

We defined a collection of user experience guidelines and best practices (in Chapter 6), and find these practices would provide a good starting point for a heuristic evaluation. Based on these new heuristics the app could be reviewed by experts in order to find general and more advanced improvements.

In order make use of already known issues a few important improvements are going to be implemented. One of the most important issues, concerns the touch recognition provided by

the Sencha framework and how to solve the performance issues. About half of the participants encountered problems with unrecognized taps. Therefore this issue will have the highest priority.

Secondly the checkout process needs to be more intuitive. Particularly the final and most important step (the actual purchase) requires further enhancement. Less important usability issues (see Table D.1) will be addressed as well, to provide further improvements for this product.

Usability tests and UX surveys can be rolled out to real users after the app is launched. This will open up a multitude of additional possibilities, including automatically generated measurements like the number of clicks and time on tasks. Statistics like that can be easily determined by analyzing data from Google Analytics.

A. Online Shopping and Tablet Statistics

Table A.1 show the rates of online shoppers in Austria published on 22.10.2012 by Statistic Austria [1] [2].

Range of Age [years]	**Percent of frequent online shoppers [%]**
16 to 24	59,5
25 to 34	70,6
35 to 44	63,8
45 to 54	45,3
55 to 64	27,0

Table A.1.: Statistic of frequent online shoppers among ages in Austria (2012)

Results had been collected for selected european countries in 2011 by Sanoma[3]. Here 26% of people between 25 and 34 own a tablet device. 31% of people between 35 to 44 aged people own and 18% between 45 to 54. Details are shown in Table A.2

Range of Age [years]	**Users of iPads [%]**
16 to 24	10
25 to 34	26
35 to 44	31
45 to 54	18
55 to 64	9

Table A.2.: Statistic of iPad user in selected european countries (2011)

[1] http://news.lindeonline.at/archives/7053-Statistik-Online-Shopping-in-OEsterreich-auf-dem-Vormarsch.html

[2] http://www.statistik.at/web_de/statistiken/informationsgesellschaft/ikt-einsatz_in_haushalten/022212.html

[3] http://de.statista.com/statistik/daten/studie/203080/umfrage/altersverteilung-der-ipad-nutzer-in-europa/

B. AttrakDiff UX Questionnaire

The AttrakDiff Questinaire by Hassenzahl is available online under `www.attrakdiff.de`

AttrakDiff Item Left	**7 Point Likert Scale**							**AttrakDiff Item Right**	**Dimension**
human	O	O	O	O	O	O	O	technical	PQ
isolating	O	O	O	O	O	O	O	connective	HQ-I
pleasant	O	O	O	O	O	O	O	unpleasant	ATT
inventive	O	O	O	O	O	O	O	conventional	HQ-S
simple	O	O	O	O	O	O	O	complicated	PQ
professional	O	O	O	O	O	O	O	unprofessional	HQ-I
ugly	O	O	O	O	O	O	O	attractive	ATT
practical	O	O	O	O	O	O	O	impractical	PQ
likeable	O	O	O	O	O	O	O	disagreeable	ATT
cumbersome	O	O	O	O	O	O	O	straightforward	PQ
stylish	O	O	O	O	O	O	O	tacky	HQ-I
predictable	O	O	O	O	O	O	O	unpredictable	PQ
cheap	O	O	O	O	O	O	O	premium	HQ-I
alienating	O	O	O	O	O	O	O	integrating	HQ-I
brings me closer to people	O	O	O	O	O	O	O	separates me from people	HQ-I
unpresentable	O	O	O	O	O	O	O	presentable	HQ-I
rejecting	O	O	O	O	O	O	O	inviting	ATT
unimaginative	O	O	O	O	O	O	O	creative	HQ-S
good	O	O	O	O	O	O	O	bad	ATT
confusing	O	O	O	O	O	O	O	clearly structured	PQ
repelling	O	O	O	O	O	O	O	appealing	ATT
bold	O	O	O	O	O	O	O	cautious	HQ-S
innovative	O	O	O	O	O	O	O	conservative	HQ-S
dull	O	O	O	O	O	O	O	captivating	HQ-S
undemanding	O	O	O	O	O	O	O	challenging	HQ-S
motivating	O	O	O	O	O	O	O	discouraging	ATT
novel	O	O	O	O	O	O	O	ordinary	HQ-S
unruly	O	O	O	O	O	O	O	manageable	PQ

Table B.1.: AttrakDiff Questionnaire

C. Simple Usability Scale Questionnaire

Table C.1 shows the 10 questions of the SUS questionnaire.

# Item	SUS	Question
1		I think I would like to use this website application frequently
2		I found the website unnecessarily complex
3		I thought the website was easy to use
4		I think I would need Tech Support help to use this website
5		I found the various functions in the website to be well integrated
6		I thought there was too much inconsistency in this website
7		I believe that most people would learn to use this website very quickly
8		I found the website very cumbersome to use
9		I felt very confident in using the website
10		I need to learn a lot about this website before I could effectively use it

Table C.1.: Simple Usability Scale Questionnaire

An optional comment could be made by the end of the SUS questionnaire. Table C.2 shows these annotations.

ID	Comment	Version
1	Good overall impression!	Tablet
2	good overall onlineshop	Desktop
3	Too much Orange	Tablet
4	More simpl, less overloaded	Tablet
5	Subjektiv slower website	Desktop
6	Sexy	Tablet
7	The Navigation sucks	Tablet
8	Nice Colors	Web
9	Ugly Colors	Tablet
10	App seems to have less functionality :(	Tablet
11	Navigation Elements well structured and positioned	Tablet

Table C.2.: Comments of Users after the SUS questionnaire

D. List of Usability Issues

Issue ID	Name	System	Type	Severity
1	AGB checkbox at registration	Desktop	Bug	High
2	Reserve instead availability	Desktop	U-Issue	Low
3	Hard to find Storesearch	Desktop	U-Issue	Medium
4	Stores not accessible via search	Desktop/Tablet	U-Issue	Medium
5	Delete Item in recall list	Desktop	Bug	Medium
6	Location Button in Storesearch	Tablet	U-Issue	High
7	Storesearch only PLZ	Tablet	U-Issue	Medium
8	Route to Store/Timetable for Train	Tablet	U-Issue	Low
9	Tap not recognized	Tablet	U-Issue	Medium
10	PLZ not found	Desktop	Bug	Low
11	Errormessage for Password	Tablet	Bug	Medium
12	Order Confirmation	Tablet	U-Issue	Medium
13	Storesearch failure	Tablet	Bug	High
14	To small input elements	Desktop/Tablet	U-Issue	Medium
15	Storesearch hard to find	Desktop	U-Issue	Low
16	Submenue not intuitiv	Tablet	U-Issue	Medium
17	Melectronics logo not touchable	Tablet	U-Issue	Medium
18	Password only once	Tablet	U-Issue	Low
19	Login / Registrieration mixed up	Tablet	U-Issue	Low
20	Dropdown menu are annoying	Desktop	U-Issue	Medium
21	Merge Produkt schwer ersichtlich	Desktop	U-Issue	Low
22	No swipe for product teaser	Desktop	U-Issue	Low
23	Search for store "Appenzell"	Desktop	Bug	Low
24	App reloads when recalled	Tablet	U-Issue	Medium
25	Screener not touchable	Tablet	U-Issue	Low
26	Network issue	Tablet	-	
27	"Computer Peripherie" category name	Desktop/Tablet	U-Issue	Low
28	Password to short	Tablet	-	
29	Registration no loading spinner	Tablet	U-Issue	Low
30	To less feedback for shopselection	Tablet	U-Issue	Low
31	No AGB link	Tablet	U-Issue	Medium
32	Address data form: name	Tablet	U-Issue	Low
33	Address data form: plz -> city	Tablet	U-Issue	Low
34	scroller at registration	Tablet	U-Issue	Low
36	wait list mixed up with recall list	Desktop	U-Issue	Low
37	cumulus not explained	Tablet/Desktop	U-Issue	Medium
38	"Themenbühne" content	Tablet	U-Issue	Low
39	scrolling in checkout is weird	Tablet	U-Issue	Low
40	product overview is showing to much content	Desktop	U-Issue	Low

Table D.1.: Usability Issues and Bugs found during the study, Part 1

Issue ID	Name	System	Type	Severity
41	screen for addon services is confusing	Desktop/Tablet	U-Issue	Medium
42	term "verfügbarkeit"	Tablet	U-Issue	Medium
43	filialreservierung overlay nicht gut designed (verfügbarkeit schlcht positioniert)	Tablet	U-Issue	Medium
44	"Heimlieferung" unknown	Desktop	U-Issue	Low
45	next step button	Desktop	U-Issue	Low
46	storereservation is confusing	Desktop	U-Issue	Low
47	store overlay to big	Desktop	layout	Low
48	product preview bug	Tablet	Bug	High
49	bug on site when preview of product is closed	Tablet	Bug	High
50	Konto button for registration	Desktop/Tablet	U-Issue	Medium
51	Too much to write	Tablet/Desktop	U-Issue	Medium
52	wrong format for Star	Tablet	layout	Low
53	Layout for Errormessage	Tablet	layout	Low
54	Submition of Order is confusing	Tablet	U-Issue	High
55	inconsitent nameing	Tablet	inkonsitenz	Low
56	verservierung/verfügbarkeit	Tablet/Desktop	U-Issue	Low
57	registration view is cluttered	Desktop	U-Issue	Medium

Table D.2.: Usability Issues and Bugs found during the study, Part 2

E. Frequency of Usability Issues

Issue ID	Frequency	Version	Issue ID	Frequency	Version
9	17	Tablet	21	1	Desktop
20	9	Desktop	23	1	Desktop
7	8	Tablet	24	1	Tablet
14	8	Beide	28	1	Desktop
8	7	Tablet	29	1	Tablet
5	6	Desktop	30	1	Tablet
6	6	Tablet	31	1	Tablet
25	5	Beide	33	1	Tablet
17	4	Tablet	36	1	Tablet
3	3	Desktop	38	1	Tablet
12	3	Tablet	39	1	Tablet
16	3	Tablet	42	1	Tablet
37	3	Beide	43	1	Tablet
41	3	Beide	44	1	Desktop
56	3	Beide	45	1	Desktop
1	2	Desktop	46	1	Desktop
10	2	Desktop	47	1	Desktop
11	2	Tablet	48	1	Tablet
22	2	Desktop	49	1	Tablet
27	2	Beide	52	1	Tablet
32	2	Beide	53	1	Tablet
40	2	Desktop	54	1	Tablet
50	2	Beide	55	1	Tablet
51	2	Beide	57	1	Desktop
4	1	Beide	15	1	Desktop
13	1	Tablet	18	1	Tablet
19	1	Tablet			

Table E.1.: Frequency of Usability Issues

F. Task Completion Times

Task #	Desktop		Tablet	
	Mean Time	**Std. Dev.**	**Mean Time**	**Std. Dev.**
1	00:03:26	00:01:37	00:03:27	00:01:44
2	00:03:59	00:01:50	00:04:17	00:02:07
3	00:02:40	00:01:35	00:02:19	00:01:13
4	00:02:21	00:01:17	00:02:34	00:01:21
5	00:02:18	00:01:04	00:01:55	00:01:17

Table F.1.: Task Completion Time - Means and Standard Deviations

G. AttrakDiff Results

The following table shows the summarized mean AttrakDiff score-values. The values represent Pragmatic Quality (PQ), Hedonic Quality - Stimulation (HQ-S), Hedonic Quality - Identity (HQ-I) and Attractiveness (ATT). Due to the result report neither of the differences is statistically significant.

Version	**PQ**	**HQ-I**	**HQ-S**	**ATT**
Desktop	1,46	1,29	-0,41	1,52
Tablet	1,51	0,81	-0,13	1,43

Table G.1.: Mean Values from the AttrakDiff questionnaire

A more fine grained visualization of single word pairs, extracted from the AttrakDiff summary report is shown in Figure G.1

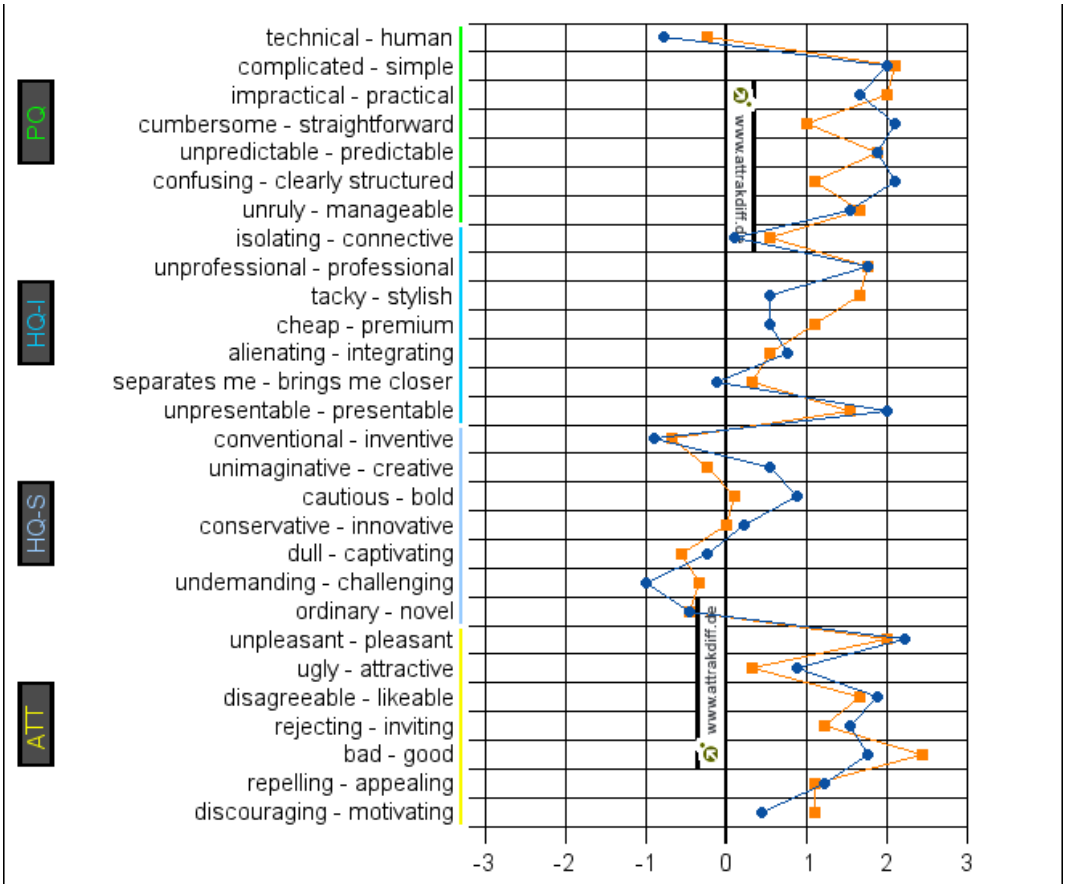

Figure G.1.: AttrakDiff Wordpairs. Orange: Desktop and Blue: Tablet Version.

List of Abbreviations

W3C	World Wide Web Consortium
WHATWG	Web Hypertext Application Technology Working Group
OS	Operating System
SASS	(Syntactically Awesome Style Sheets
CDN	Content Delivery Network
CSS	Cascading Style Sheets
UI	User Interface

Bibliography

[1] "IDC - Press Release: Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter." http://www.idc.com/getdoc.jsp?containerId=prUS23771812, Date accessed: 2013-03-13.

[2] "IDC - Press Release: Android Tablets Gain Momentum in the Third Quarter, Expectations Remain High for the Holiday Quarter." https://www.idc.com/getdoc.jsp?containerId=prUS23772412, Date accessed: 30-11-2012.

[3] M. Kenney and B. Pon, "Structuring the Smartphone Industry: Is the Mobile Internet OS Platform the Key?," *Journal of Industry, Competition and Trade*, vol. 11, pp. 239–261, June 2011.

[4] H. Lee and E. Chuvyrov, *Beginning Windows Phone App Development*. Apress, 2012.

[5] R. Hinman, *The Mobile Frontier*. Rosenfeld Media, 2012.

[6] W. A. Thomas Tullis, *Measuring the User Experience*. Morgan Kaufmann, 2008.

[7] "Global Tablet OS Market Share: Q3 2012." http://www.businesswire.com/news/home/20121025006932/en/Strategy-Analytics-Android-Captures-Record-41-Percent, Date accessed: 2013-03-03.

[8] J. Nielsen, *Usability Engineering*. Morgan Kaufmann Series in Interactive Technologies, Acad. Press, 1994.

[9] C. Fernandez and C. Kühn, "Mobile Web & Responsive Webdesign," *i-com*, vol. 11, pp. 53–56, Mar. 2012.

[10] "Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth." http://www.gartner.com/it/page.jsp?id=1924314, Date accessed: 2012-12-03, 2012.

[11] B. Fling, *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps*. O'Reilly Media, Inc., 2009.

[12] IDC, "IDC Forecasts Worldwide Tablet Shipments to Surpass Portable PC Shipments in 2013, Total PC Shipments in 2015." http://www.idc.com/getdoc.jsp?containerId=prUS24129713, Date accessed: 2013-06-13.

[13] "Gartner Says Worldwide PC, Tablet and Mobile Phone Combined Shipments to Reach 2.4 Billion Units in 2013." http://www.gartner.com/newsroom/id/2408515, Date accessed: 2013-06-14.

[14] S. Lerotic, "Distributing Applications in 2012," tech. rep., University of Trieste, 2012.

[15] F. Nayebi, "The state of the art of mobile application usability evaluation," *IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 25, pp. 1–4, 2012.

[16] K. Puputti, *Mobile HTML5 : Implementing a Responsive Cross-Platform Application*. Masters thesis, Aalto University, 2012.

[17] T. Rauche, "Summative usability evaluation: Hedonic and pragmatic quality of a mobile device application.." 2010.

[18] A. Kaikkonen, A. Kekäläinen, M. Cankar, T. Kallio, and A. Kankainen, "Usability Testing of Mobile Applications : A Comparison between Laboratory and Field Testing," *Journal of Usability Studies*, vol. 1, no. 1, pp. 4–16, 2005.

[19] J. Nielsen, L. A. Blatt, J. Bradford, and P. Brooks, "Usability Inspection Methods," in *Conference companion on Human factors in computing systems CHI 95*, vol. 25, pp. 413–414, ACM Press, 1994.

[20] C. Coursaris and D. Kim, "A meta-analytical review of empirical mobile usability studies," *Journal of Usability Studies*, vol. 6, no. 3, pp. 117–171, 2011.

[21] J. Kwahk and S. Han, "A methodology for evaluating the usability of audiovisual consumer electronic products," *Applied Ergonomics*, vol. 33, pp. 419–431, 2002.

[22] J. Mrazova, *User Centered Cross-Platform Application Development for Mobile Devices*. PhD thesis, University of Applied Sciences, Vienna, 2012.

[23] B. Browne, "Five Lessons from a Year of Tablet UX Research." `http://uxmag.com/articles/five-lessons-from-a-year-of-tablet-ux-research`, Last accessed: 2012-09-09.

[24] S. Pekkala, *Usability evaluation of design solutions for tablet magazines*. PhD thesis, Aalto University, 2012.

[25] I. Hawk Partners, "Mobile and Tablet e-Commerce : Is Anyone Really Ready ?." 2012.

[26] R. Thampy;, P. Agrawal, S. Mandal, K. Chaitanya, N. Venkataraman, C. Gokhale, V. Swaminathan, and J. Gohel, "Advances in User Experience Design," Tech. Rep. 1, Infosys, 2012.

[27] A. B. Naumann and I. Wechsung, "Developing Usability Methods for Multimodal Systems: The Use of Subjective and Objective Measures," *Proceedings of the International Workshop on Meaningful Measures*, 2008.

[28] IDC, "Android and iOS Surge to New Smartphone OS Record in Second Quarter." `http://www.idc.com/getdoc.jsp?containerId=prUS23638712`, Date accessed: 01-01-2013.

[29] R. Rogers, J. Lombardo, Z. Mednieks, and B. Meike, *Android Application Development*. O'Reilly Media, Incorporated, 2009.

[30] IDC, "Strong Apple Shipments Drive Robust Tablet Market Growth in Second Quarter." `http://www.idc.com/getdoc.jsp?containerId=prUS236325120`, Date accessed: 2012-09-10.

[31] M. Firtman, *Programming the Mobile Web, 2nd Edition*. O'Reilly Media, 2nd editio ed., 2012.

[32] "Mobile / Tablet Operating System Market Share." `http://www.netmarketshare.com/`, Date accessed: 2012-12-18, 2012.

[33] J. Ohrt and V. Turau, "Cross-platform development tools for smartphone applications," pp. 72–79, 2012.

[34] R. Padley, "HTML5 - bridging the mobile platform gap : mobile technologies in scholarly communication," *Serials: The Journal for the Serials Community*, vol. 24, no. November, pp. S32–S39, 2011.

[35] A. Charland and B. LeRoux, "Mobile application development: web vs. native," *Communications of the ACM*, pp. 49–53, 2011.

[36] B. Lawson and R. Sharp, *HTML5*. Addison-Wesley Verlag, 2012.

[37] M. Firtman, *Programming the Mobile Web*. O'Reilly Media, 2012.

[38] Telerik, "Kendo UI." `http://www.kendoui.com/`, Date Accessed: 2013-07-12.

[39] W3C, "HTML5 Performance - W3C Wiki." `http://www.w3.org/wiki/HTML5_Performance`, Date Accessed: 2013-07-13, 2013.

[40] Strategy Analytics, "Firefox OS smartphone forecast." `http://blogs.strategyanalytics.com/WSS/post/2012/09/27/Firefox-OS-to-Capture-1-Percent-Share-of-Global-Smartphone-Market-in-2013.aspx`, Date accessed: 201-08-253.

[41] R. Müller, B. Kijl, and J. Martens, "A Comparison of Inter-Organizational Business Models of Mobile App Stores: There is more than Open vs. Closed," *Journal of theoretical and applied electronic commerce research*, vol. 6, no. 2, pp. 13–14, 2011.

[42] R. Holly, *Taking your Android Tablets to the max*. Apress, 2012.

[43] Sights, "The HTML5 test." `http://html5test.com/results/mobile.html`, Date accessed: 2012-12-20.

[44] G. Frederick, *Beginning Smartphone Web Development: Building JavaScript, CSS, HTML and Ajax-based Applications for iPhone, Android, Palm Pre, BlackBerry, Windows Mobile, and Nokia S60*. Apress, 2010.

[45] M. Staikos, "BlackBerry 10 Browser Meets Benchmark Standard for HTML5 Apps -BlackBerry Developer Blog." `http://devblog.blackberry.com/2012/11/blackberry-10-browser-html5-standards/`, Date accessed: 2012-12-02.

[46] M. Firtman, *jQuery Mobile: Up and Running*. O'Reilly Media, Inc., 2012.

[47] "PC Market in Western Europe Declined 15 Percent in the Third Quarter of 2012." `http://www.gartner.com/it/page.jsp?id=2230815`, Date accessed: 2013-01-13.

[48] J. Nielsen, *Mobile Usability*. New Riders, 2012.

[49] W3C, "HTML5 Specification - Editor's Draft 21 November 2012." `http://dev.w3.org/html5/spec/single-page.html`, Date accessed: 2012-11-23.

[50] "WHATWG - HTML Living Standard." `http://www.whatwg.org/specs/web-apps/current-work`, Date accessed: 2013-05-21.

[51] A. van Kesteren and M. Stachowiak, "HTML Design Principles." `http://www.w3.org/TR/html-design-principles/`, Date accessed: 2012-11-02.

[52] P. F. von Rotenhan, "Usability von Web-Anwendungen - mobile Endgeräte," 2012.

[53] Z. Mednieks, L. Dornin, G. B. Meike, and M. Nakamura, *Programming Android*. O'Reilly Media, Inc., 2nd editio ed., 2012.

[54] S. Komatineni and D. MacLean, *Pro Android 4*. Apress, 2012.

[55] A. Nahavandipoor, *iOS 6 Programming Cookbook*. O'Reilly Media, Inc., 2012.

[56] S. Allen, V. Graupera, and L. Lundrigan, *Pro Smartphone Cross-Platform Development*. Apress, 2010.

[57] Lionbridge, "Mobile Web Apps vs . Mobile Native Apps : How to Make the Right Choice." 2012.

[58] Manning, "Sencha Touch in Action." 2012.

[59] Kumar A, *Sencha MVC Architecture*. Packt Publishing, 2012.

[60] "Mobile App Development Platform | Sencha Touch." `http://www.sencha.com/products/touch`, Date accessed: 2013-02-22.

[61] "jQuery: The Write Less, Do More, JavaScript Library." `http://jquerymobile.com/`, Date accessed: 2012-12-02.

[62] K. Hadlock, *jQuery Mobile: Develop and Design: Safari Books Online*. Peachpit Press, 2012.

[63] L. Wroblewski, *Mobile First*. A Book Apart, 2011.

[64] Gartner, "Gartner Recommends a Hybrid Approach for Business-to-Employee Mobile Apps." `http://www.gartner.com/newsroom/id/2429815`, Date accessed: 2013-08-14.

[65] "Design | Android Developers." `http://developer.android.com/design/index.html`, Date Accessed: 2013-04-16.

[66] "Develop | Android Developers." `http://developer.android.com/develop/index.html`, Date accessed: 2013-05-19.

[67] Apple Inc., "iOS Human Interface Guidelines." `http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html`, Date accessed: 02-04-2013, 2013.

[68] "ELEKS Labs: Android vs. iOS: UI/UX Differences." `http://www.elekslabs.com/2012/12/android-vs-ios-user-experience.html`, Date accessed: 2013-05-09.

[69] N. Bevan, "What is the difference between the purpose of usability and user experience evaluation methods?," in *Workshop on User Experience Evaluation Methods in Product Development during Interact09 Conference*, 2009.

[70] Y. Rogers, H. Sharp, and J. Preece, *INTERACTION DESIGN: beyond human-computer interaction*. John Wiley and Sons, 3rd ed., 2011.

[71] B. Hörmannsdorfer, "Usability mobiler Webseiten," 2012.

[72] C. Barnum, *Usability Testing Essentials*. Elsevier, 2010.

[73] N. Bevan, "International standards for HCI," *Encyclopedia of human computer interaction*, no. May, pp. 1–15, 2006.

[74] N. Bevan, "Common Industry Specification for Usability-Requirements," *Technology*, no. June, 2007.

[75] N. Bevan, "ISO/IEC JTC1/SC7 N4098," 2008.

[76] P. G. Zimmermann, *Beyond Usability - Measuring Aspects of User Experience*. Dissertation, ETH Zurich, Jan. 2008.

[77] M. Hassenzahl and N. Tractinsky, "User experience - a research agenda," *Behaviour & Information Technology*, vol. 25, pp. 91–97, Mar. 2006.

[78] L. Alben, "Quality of Experience - Defining the criteria for effective interaction design," *interactions*, pp. 11–15, 1996.

[79] R. Hartson and P. S. Pyla, *The UX Book*. Morgan Kaufmann, 2012.

[80] N. Bevan, "Classifying and Selecting UX and Usability Measures," *Proceedings of the International Workshop on Meaningful Measures*, 2008.

[81] E. Law, "The measurability and predictability of user experience," *Proceedings of the 3rd ACM SIGCHI symposium on ...*, 2011.

[82] S Krug, *don't make me think!* New Riders, 2005.

[83] "Usability for Mobile Devices." `http://www.uxmatters.com/mt/archives/2010/09/usability-for-mobile-devices.php`, Last accessed: 2012-08-02.

[84] J. Sauro and James R Lewis, *Quantifying the User Experience*. Morgan Kaufmann, 2012.

[85] J. Lumsden, *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*. IGI Global, 2011.

[86] J. Nielsen, "Ten usability heuristics," 2005.

[87] N. Bevan, "Extending Quality in Use to Provide a Framework for Usability Measurement," vol. 11, no. 1998, pp. 13–22, 2009.

[88] J. Brooke, "SUS-A quick and dirty usability scale," *Usability evaluation in industry*, 1996.

[89] J. Wang and S. Senecal, "Measuring perceived website usability," *Journal of Internet Commerce*, vol. 16933, 2007.

[90] K. Väänänen-Vainio-Mattila, V. Roto, and M. Hassenzahl, "Towards Practical User Experience Evaluation Methods," *VUUM*, 2008.

[91] M. Hassenzahl, M. Burmester, and F. Koller, "AttrakDiff : Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität 1 Einleitung Grundannahmen und Vorarbeiten," *Mensch & Computer 2003*, pp. 187–196, 2003.

[92] B. Laugwitz, T. Held, and M. Schrepp, "Construction and evaluation of a user experience questionnaire," *HCI and Usability for Education and Work*, pp. 63–76, 2008.

[93] M. Beccari and T. Oliveira, "A Philosophical Approach about User Experience Methodology," in *Design, User Experience, and Usability. Theory, Methods, Tools and Practice*, vol. 6769 of *Lecture Notes in Computer Science*, pp. 13–22, Springer Berlin Heidelberg, 2011.

[94] J. Sauro and J. Dumas, "Comparison of three one-question, post-task usability questionnaires," *Proceedings of the 27th international conference on Human factors in computing systems CHI 09*, pp. 1599—-1608, 2009.

[95] T. S. Tullis and J. N. Stetson, "A Comparison of Questionnaires for Assessing Website Usability," *Usability Professional Association Conference*, pp. 1—-12, 2004.

[96] Industry Usability Reporting project, "Common Industry Format for Usability Test Reports," 1999.