Phil Diegmann

**Bachelorarbeit
im Fach Allgemeine Wirtschaftsinformatik**

Systematic Development of mHealth Apps: Lessons Learned During Development of a Mobile Frontend for ePill

Themensteller: Jun.-Prof. Dr. Ali Sunyaev

Vorgelegt in der Bachelorprüfung
im Studiengang Wirtschaftsinformatik
der Wirtschafts- und Sozialwissenschaftlichen Fakultät
der Universität zu Köln

Köln, August 2013

Table of Contents

Index of Abbreviations

API	Application Programming Interface. It specifies how software components could interact with each other.
app	Application
app user	intended audience for the app
CDN	Content Delivery Network. Multiple servers which are globally distributed for serving static content with high availability and performance.
CSS	Cascading Style Sheets. A language used to style web pages
DNS	Domain Name System. Used to translate domain names into IP-Addresses
eHealth	"a paradigm involving the concepts of health, technology, and commerce, with commerce and technology as tools in the service of health"[1]. eHealth belongs to the field of telehealth.[2]
ePill	a patient-centered health IT service which offers information on pharmaceuticals and aggregation of data in context[3]
framework	can contain source code, tools and libraries, which together provide specific or common but abstracted functionality.
frontend	visible user interface for the app user
HECAT	Health Education Curriculum Analysis Tool[4]
HIT	Health Information Technology
HTML	HyperText Markup Language, a markup language to design web pages.
IDE	Integrated Development Environment
JSON	JavaScript Object Notation, represents data structures
mHealth	"medical and public health practice supported by mobile devices, such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices"[5], also known as m-Health.

---

[1] Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013), p. 2

[2] cf. Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013), p. 2

[3] cf. Dehling, Sunyaev (2012b), p. 2

[4] http://www.cdc.gov/HealthyYouth/HECAT/

[5] World Health Organization (2011) cited by Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013), p. 2

mHealth apps	"aim at providing seamless, global access to tailored health IT services and have the potential to alleviate global health burdens"[6]
MVC	Model-View-Controller. A software architecture pattern which separates logic and user interfaces. Models are representatives of data structures. Views contains the user interface definitions and controllers contains the application logic.[7]
NDK	Native Development Kit. Bundled software and tools which enables the developer to implement programs on native-code languages.[8]
OS	Operating System
SDK	Software Development Kit. Bundled software and tools for developing with or for a specified OS or framework.
telehealth	delivery of medical- or health-related information or services via telecommunication technologies.
usability	"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"[9]
use value	the utility of consuming a good or service
user interface	for humans visible controls and layout of an application
W3C	World Wide Web Consortium[10]

---

[6]   Dehling, Sunyaev (2013), p. 1

[7]   cf. Hasan, Isaac (2011) p. 418

[8]   cf. http://developer.android.com/tools/sdk/ndk/index.html

[9]   Yeh, Fontenelle (2012), p. 64 as quoted from ISO 9241-11 (1998)

[10]   http://www.w3.org

Index of Tables

Index of Illustrations

1. Introduction

1.1 Research Problem

While it has become easy to develop a mobile health (mHealth) application (app), there is much more to it than just the aspects of the app's core functionality. Currently only very few guidelines, best practices and systematic development approaches for mobile app development can be found. Furthermore even less can be found for the specific area of mHealth apps.

Security leaks or even abuse of private and sensitive information[11] can lead to great harm for the app user and to legal issues for the developer. Abuse of personal health related information can result in loss of reputation (e.g. sexual transmitted diseases) or financial drawbacks and decreased chances of employment (e.g. chronic diseases, genetic dispositions)[12]. With poorly developed apps, there is a danger of security leaks and hence for data abuse. Thus the risk for app users increases. A study has shown that very few mHealth apps entail little or low risk for the app user.[13] Self-publishing through modern sales channels such as Google Play[14] or the iOS App Store [15] and the availability of easy-to-use Integrated Development Environments (IDEs) lower the barriers for entry. Even one-man developers or small teams are now able to publish apps with less development effort than a few years ago.[16, 17] Without fundamental knowledge of privacy and security aspects, there is an increase in the non-professional developmental of mobile apps with possibly inadequate security aspects.

Usability as well as the overall app quality is also a possibly undervalued aspect in non-

---

[11]   information, which is personal. Can be related to financial-, health- or otherwise personal relevant information, suggested by Future of Privacy Forum, Center for Democracy & Technology (2011), p. 6, although the definition varies

[12]   cf. Dehling, Sunyaev (2013), pp. 6-7

[13]   cf. Njie (2013), pp. 19-20

[14]   http://play.google.com, last visited on 09/02/2013

[15]   http://appstore.com, last visited on 09/02/2013

[16]   cf. for this and the next sentence Dehling, Sunyaev (2013), p. 2

[17]   cf. for this and the next sentence Moore et al. (2012), p. 15

professional developments.[18, 19] While fancy colors might look appealing to the developer himself, it might lead to confusion for the app user or even to a lack of operability for visually impaired people.[20] Also, the need for a intuitive user interface[21] is not be considered as important as it should be.

Knowledge of data privacy acts and laws is a premise for a legal, safe and fair development for the developer and the app user. Multiple layers of data privacy laws in Europe on international, national and state level require a certain legal knowledge.[22] Also, the benefit of and the need for a privacy policy seems to be ambiguous for many non-professional developers.[23]

This lack of guidelines for mobile app development and of specific guidelines for privacy and usability sensitive apps is only superficially considered by most of the literature. The beforehand highlighted aspects of usability and information security[24] are just two of multiple possible requirements. Current research seems not to state which specific requirements, if any, distinguish mHealth apps from other apps or which are needed to be more accented.

1.2   Objectives of this Thesis

The purpose of this thesis is to discover, identify and report issues and challenges of the development of mHealth apps by developing a mobile frontend for the ePill system[25] (developed by the University of Cologne). ePill is a patient-centered health IT service which

---

[18]   cf. Dehling, Sunyaev (2013), p. 2

[19]   cf. Mayer, Weitzel (2012), p. 1681

[20]   cf. Badashian et al. (2008) p. 108

[21]   for humans visible controls and layout of an application

[22]   cf. Directive 95/46 of the European Parliament and of the Council (October, 24th 1995), Directive 2002/58 of the European Parliament and of the Council (July, 12th 2002) cited by Future of Privacy Forum, Center for Democracy & Technology (2011), p. 16

[23]   cf. Njie (2013), p. 20

[24]   information security stands for prevention from unauthorized access, modification, use or disruption of information and information systems

[25]   http://epill.uni-koeln.de, last visited on 09/24/2013

offers information on pharmaceuticals and aggregation of pharmaceutical data in context.

During the development of a mobile frontend for ePill, all requirements can be addressed more easily than in a completely theoretical context. A mobile app for ePill will increase the accessibility for the ePill system in general, and thereby increase the possible user value. In critical situations in which one does not have a desktop computer at hand, a mobile easy-to-use app can be of value. It is important to us, that the developed app is at least accessible on Android and iOS devices.

The experiences gained during the development refer to general mobile app development, but also to the specific development of mHealth apps.

Mainly this thesis aims to describe the planning and the development process and discuss all discovered issues and challenges for planning and developing mHealth apps. One sub-objective is to give a short overview about the state of research on guidelines and important factors of mHealth app development. We will validate the developed app by reference to the stated guidelines and best practices. Also we aim to provide an short overview over different frameworks, compare them and give a short evaluation about the framework we chose for the implementation of the mobile frontend for ePill.

2. The ePill System

2.1 The System in General

The ePill system (http://epill.uni-koeln.de) was developed by the University of Cologne to improve the readability and comprehensibility of instruction leaflets contained within the packaging of pharmaceutical drugs. Additionally ePill aims to provide further information on adverse reactions and interactions of different medical drugs. ePill emphasizes an easy readability and access to informations.

ePill is currently a prototype of the planned system, used only for research purposes and it is only actively used by the University of Cologne. Therefore it is only localized in German and contains just pharmaceuticals available in Germany. ePill utilizes the "GELBE LISTE PHARMINDEX"[26], provided by Medizinische Medien Informations GmbH MMI as main source for pharmaceutical information.

The system provides three major functions: searching for pharmaceuticals, display information on pharmaceuticals and supplementing services.[27] The search enables the user to find corresponding pharmaceuticals depending on specified parameters in the underlying database. As an extend, the display functionality enables the user to read the leaflet information in an optimized fashion. Finally supplementing services are provided to refine the displayed information (e.g. select the level of detail of the displayed information). Comparing pharmaceuticals side by side as well as a list of adverse reactions or side effects can be displayed for a list of pharmaceuticals.

An integration and personalization depending on the current user's health records was not implemented due to the arising privacy and trust challenges.[28, 29] Instead the system was focused on patient friendliness regarding readability and comprehensibility.[30]

---

[26]   http://www.gelbe-liste.de, last visited on 09/01/2013

[27]   cf. for this paragraph Dehling, Sunyaev (2012a), p. 2 and Dehling, Sunyaev (2012b), p. 5

[28]   cf. Kaletsch, Sunyaev (2011) cited by Dehling, Sunyaev (2012a), p. 2

[29]   cf. Kaletsch, Sunyaev (2011), pp. 5-6

[30]   cf. Dehling, Sunyaev (2012b), p. 2

The system uses a Model-View-Controller (MVC) architecture[31] and utilizes a relational database as persistent data storage.[32] The data is organized with the pharmaceutical ingredients in mind. Products are any pharmaceuticals, which may contain specific molecules, which themselves may be related to specific adverse reactions with other molecules.

With this atomized organization of the pharmaceutical information, it becomes more easier to compare different pharmaceuticals and have very consistent information about molecules and adverse reactions with other pharmaceuticals.

2.2   The Web Application

The web application of the ePill system introduces itself highly customizable to the user. It offers the user to choose between a default view, a customizable view and an expert view. The default view aims to provide all necessary information in a compact way. The customizable view offers more choices for the controls and user interface elements to be displayed. The expert view activates the most detailed information level. The pharmaceutical informations to be displayed can be fine tuned for every view. ePill offers four different presets varying from only the most basic up to all available information. These presets can be further customized by afterwards selecting or deselecting items. ePill offers customization for the displayed leaflet information as well as the visibility of controls for access to supplementing services and settings in the sidebar. Additionally the font-size can be set to normal, bigger and biggest to support visually impaired users.

Three columns shape the layout. The first, leftmost column contains the main navigation for searching, pharmaceutical listings, basic functionality such as help pages and settings as well as extended functionality such as interactions research and adverse reaction lookup or pharmaceutical comparisons. The second, centered column contains the current content. This column has tabs[33], which can be assigned different contents. With this tabular layout, e.g. multiple, different search queries can easily be switched and held in parallel. The third, rightmost column can be used to dynamically display or hide se-

---

[31]   cf. Dehling, Sunyaev (2012a), p. 3

[32]   cf. Dehling, Sunyaev (2012a), p. 5 for this and the following two sentences

[33]   user interface elements which imitate the look and functionality of card tabs inserted in paper files

lected information. Depending on the beforehand selected view, the left or right columns are hidden or visible. On the pharmaceutical detail page, the website offers the user the functionality to explain any term as well as a shortcut to the page's top.

The specific content layout is very consistent. Headlines are made salient and the arrangement of common sections are congruent. Changes in settings are applied with no delay and without a page reload.

Although this web application is not optimized for mobile applications and designed with a desktop computer in mind, it can be accessed by nearly any modern mobile computing device, such as a smart phone or a tablet, and can therefore categorized as a mHealth application. This categorization is important to the following section, to clarify the differences between the web application and the mobile client.

3. What is mHealth?

3.1 Definition

eHealth is defined as "a paradigm involving the concepts of health, technology, and commerce, with commerce and technology as tools in the service of health"[34].

Telehealth stands for the delivery of medical- or health-related information or services via telecommunication technologies.

mHealth in detail was defined as "medical and public health practice supported by mobile devices: such as mobile phones, patient monitoring devices, personal digital assistants (PDAs), and other wireless devices"[35]. The introduction of smart phones such as the Apple iPhone and Android devices led to a greater audience and the evolution of mobile tablets further increasing the audience for mHealth purposes. A study relied on the Health Education Curriculum Analysis Tool (HECAT)[36] to group different mHealth apps together.[37]. HECAT content areas are describing eHealth application content areas, not only content areas for mHealth apps. The study by West et al. (2012) illustrates the distribution of apps in the HECAT content categories listed in Tab. 3-1.[38] As Tab. 3-1 illustrates most of the available apps in 2011 in the Apple App Store in the United States of America belonged to the Physical Activity area, whereas drug-related and safety-related apps (like ePill) are the least two categories in terms of magnitude.

From February to May of 2012, a study by d'Heureuse et al. (2012) found several ten thousands of apps in the Google Play Store as well as the Apple App Store limited to the "Health" categories.[39] The study by d'Heureuse et al. (2012) shows the potential of mHealth for a broader healthcare supported by mobile devices. From March to May of 2012, the total number of apps increased by an average of 6.4% (Google Play Store) and

---

[34] Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013), p. 2

[35] World Health Organization (2011) cited by Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013), p. 2

[36] http://www.cdc.gov/HealthyYouth/HECAT/, last visited on 09/07/2013

[37] cf. West et al. (2012)

[38] cf. West et al. (2012)

[39] cf. d'Heureuse et al. (2012), p. 20, Figure 5

4.5% (Apple App Store) per month.[40]

HECAT content area	n	%[41]
Physical Activity	1108	33.21
Personal health and wellness	962	28.84
Healthy eating	651	19.51
Mental and emotional health	414	12.41
Sexual and reproductive health	243	7.28
Alcohol, tobacco, and other drugs	131	3.93
Violence prevention and safety	96	2.88

Tab. 3-1: Distribution of Apps related to their HECAT Content Area (N = 3336)[42]

3.2 mHealth App Categories

Although Tab. 3-1 lists categories for mHealth apps, it focusses on content and less on the specifics for mHealth apps on other possibly important topics, such as information security or usability. Those content areas range from physical activity to health and wellness (mental, emotional, sexual or diet related) as well as for drugs and prevention. Other literature focusses on data practices and privacy risks with a more technical aspect[43].

Njie (2013) concludes that most of the mHealth apps deal in any way directly or indirectly (e.g. via usage behavior) with sensitive information.[44] Therefore ten levels of privacy risks were developed and a sample of 43 mHealth and fitness apps were assigned to the different levels. Tab. 3-2 illustrates the characteristics of every level as well as the distribution of the 43 analyzed apps.

The risk levels are based on the one hand on the information available to the app and on the other hand on security precautions implemented by the developer to prevent unau-

---

[40]   cf. d'Heureuse et al. (2012), p. 20

[41]   Apps could be assigned to multiple categories

[42]   cf. West et al. (2012), p. 6, Table 2

[43]   cf. Njie (2013), pp. 13-14

[44]   cf. for this and the following paragraph Njie (2013), pp. 13-14, 19, 21

thorized access to this information. An important differentiation is also in the anonymity or identifiability of the information accessible by third parties, such as other apps, other developers or unauthorized individuals. The higher the accessibility, the identifiability or the possible harm done by this information, the higher the risk level to be assigned.

Level	Risk	Characteristics	%[45]
9	Highest	address, financial information, full name, sensitive or embarrassing health (or health-related) information, information that a malicious actor could use to steal or otherwise cause a user to lose money	40
8	High	geo-location	
7	Medium-high	DOB, ZIP code, any kind of personal medical information	
6	Medium	risk evaluated to be between level 5 and level 7	32
5	Medium	email, first name, friends, interests, weight, information that is potentially embarrassing or could be used against a person (e.g., in employment)	
4	Medium	risk evaluated to be between level 5 and level 3	
3	Medium-low	anonymized (not personally identifiable) tracking (e.g., app usage), device info, a third party knows the user is using a mobile medical app	28
2	Low	risk evaluated to be between level 3 and level 1	
1	Low	any kind of anonymized data that does not include medical health-related data or personally identifiable information	
0	No		0

Tab. 3-2: Privacy Risk Levels of mHealth Apps (N = 43)[46]

As stated by Istepanian, Jovanov, Zhang (2004), another categorization is possible. They categorized mHealth applications into administrative connectivity, financial connectivity or medical connectivity.[47] Because of the lack of smart phones and a far lesser availability of mobile devices in 2004 compared to today, this article cannot take the recent

---

[45]   of apps in the sample

[46]   cf. Njie (2013), p. 13

[47]   cf. Istepanian, Jovanov, Zhang (2004), p. 409

development in mobile devices into account. The administrative connectivity handles appointments, electronic patient records and any non-financial transactions.[48] The financial connectivity includes all financial transactions such as purchases, billing or any financial services. The third connectivity, the medical connectivity, stands for mobile monitoring and diagnostics.

As described, there are three different categorization approaches for mHealth applications: The content, the information security risk-level and the overall connectivity function. For the content-category as well as the connectivity-category, multiple assignments are possible. Combined these categorization approaches form a specific grouping of mHealth apps. Depending on the categorization in the privacy risk, one can take precautionary measures. With the categorization in a HECAT content area one can identify the target audience more precisely as well as with the help of the connectivity category.

3.3 Classification of the ePill Web Application

ePill can be categorized in the HECAT content areas as "Alcohol, tobacco, and other drugs", because it informs about (medical) drugs. Since ePill also informs about adverse effects and interactions between pharmaceuticals, it belongs furthermore to the content area "Violence prevention and safety".

The ePill web application is not connected to any electronic patient records, nor does it store any user related information such as a history of the last searched pharmaceuticals, but it does not utilize SSL-encryption. Therefore it might not be collecting information or storing anything, but third parties could collect user specific information by monitoring. Putting this information into context with the risk levels developed by Njie (2013), if SSL-encryption would be utilized by the ePill web application, it could be categorized as level three. With SSL-encryption, third parties could still retrieve browser and OS specific information, but not the data sent and retrieved with each request such as pharmaceutical information. Without encryption all data sent and retrieved is visible to possible eavesdropper. With information about searched pharmaceuticals, one could assemble an overall

---

[48] cf. for this and the two following sentences Istepanian, Jovanov, Zhang (2004), p. 409

picture of the ingested medications and therefore extrapolate possible diseases. Still, all data is anonymized.

Having in mind, that ePill still is in early prototyping and assuming, that the SSL-encryption will be an upcoming feature, the risk is more of a medium to low level. Dealing with anonymous data only and protecting them with encryption leaves very little room for serious risks. We would therefore categorize ePill as a level two in terms of privacy risk levels.

Although ePill does not absolutely fit in any of the connectivity categories, its closest fit is within medical connectivity. Because of the aim to provide pharmaceutical (therefore medical) information, it could still be found within the medical connectivity category.

Concluding this categorization, we would suggest to categorize the ePill web application as a low privacy risk, drug and safety-related medical connectivity mHealth application.

3.4 Why is a special Focus on mHealth Apps warranted?

mHealth apps differ in some way from general (mobile) applications but also from eHealth applications. While mHealth apps can be used in many different situations and with very different intentions, the special focus on e.g. equality of all users and accessibility for all possible users is important for mHealth apps. This is supported by the definition of mHealth apps: "mHealth apps aim at providing seamless, global access to tailored health IT services and have the potential to alleviate global health burdens"[49], which means that they should be accessible by mostly all possible users. Whereas other types of apps do not necessarily need to be accessible by any user, we want to stress that accessibility does not only mean usability (especially for elderly people), but also global for different social layers or cultures[50].

mHealth apps deal with medical- or health-related information and have therefore to deal with sensitive information and have to address privacy risks and concerns. As pointed

---

[49]   Dehling, Sunyaev (2013), p. 1

[50]   cf. Dehling, Sunyaev (2013), p. 1

out by Njie (2013) and already referred to in Tab. 3-2, many mHealth apps deal with highly sensitive data and have serious privacy risks. Dehling, Sunyaev (2013) illustrates the possible damages through leaks, manipulation or loss of information, such as loss of affection and reputation or lessened employment possibilities.[51]

---

[51]   cf. Dehling, Sunyaev (2013), p. 7

4. The Development of the Mobile Client

4.1 Preconditions

4.1.1 Norms for Mobile Apps

Because ePill is currently used in Germany only, we will focus on laws applicable in Germany. These laws are namely the Telekommunikationsgesetz, the Telemediengesetz, the Directive 95/46/EG as well as the data protection act of North Rhine-Westphalia. The Telekommunikationsgesetz and Telemediengesetz are laws by state, whereas Directive 95/46/EG is an european directive, specified by the respective Member States.
German federal states have their own data protection acts. In this thesis we will focus on the data protection act of North Rhine-Westphalia as ePill is located in North Rhine-Westphalia.

As the topmost layer of laws, the Directive 95/46/EG defines more general directives. Article 4 defines national law applicable, if the natural or legal person, the controller[52], is located on a Member State's territory[53] or if any of the processing takes place on a Member State's territory[54]. It is required, that the controller asks the user to consent to the use and collection of data[55]. Explicitly, "data concerning health and sex life"[56] shall not be processed, only if the user consent explicitly[57], or if the processing is done by a healthcare professional under national law and for preventive medicine, medical diagnosis or treatment or for the management of health-care services[58].

This is refined by the Telemediengesetz. § 13 section (1) states, that the controller has to inform the user in a commonly understandable manner about the data which is col-

---

[52]   cf. The European Parliament and the Council of the European Union (1995), Article 2, (d)

[53]   cf. The European Parliament and the Council of the European Union (1995), Article 4, 1., (a) and (b)

[54]   cf. The European Parliament and the Council of the European Union (1995), Article 4, 1., (c)

[55]   cf. The European Parliament and the Council of the European Union (1995), Article 7, (a)

[56]   The European Parliament and the Council of the European Union (1995), Article 8, 1.

[57]   cf. The European Parliament and the Council of the European Union (1995), Article 8, 2., (a)

[58]   cf. The European Parliament and the Council of the European Union (1995), Article 8, 3.

lected and the form of processing of this data[59]. For a legal consent, the controller has to ensure, that the user is aware of his consent, that the consent is minuted, that the content of the consent is always available to the user and that the user can revoke his consent[60].

The same laws are stated in §§ 91, 93 and 94 of the Telekommunikationsgesetz[61].

Also the data protection act of North Rhine-Westphalia constitutes the same laws[62] with the only restrictions, that its scope is limited to North Rhine-Westphalia.

Therefore ePill should explicitly inform the user that no data is stored and only anonymized transacted to find matching results, to comply with the stated laws.

4.1.2 Best Practices

The World Wide Web Consortium (W3C) has published a document in 2008 which states the basic best practices for developing for the mobile web. This document states 60 best practices, which shall ensure a minimum quality level for mobile web applications. These best practices emphasize the need of regard of the device's capabilities and supported technologies[63].

The document focuses on mobile web development[64], which has differences to native app development (e.g. the usage of frames and the accessibility of the device's specific features). Most of the best practices are applicable in both development environments.

For the mobile frontend of ePill, we can focus on best practices related to the user interface, input and navigation methods as well as general best practices, because it does not need more specific device capabilities, such as positioning and navigation features. Depending on the framework chosen, some of the best practices are already implemented by the framework. I.e. a thematic consistency[65] is by default provided by frameworks

---

[59]   cf. Bundesregierung der Bundesrepublik Deutschland (2007), § 13, section (1)

[60]   cf. Bundesregierung der Bundesrepublik Deutschland (2007), § 13, section (2)

[61]   cf. Bundesregierung der Bundesrepublik Deutschland (1996), Section 2, §§ 91, 93, 94

[62]   cf. Der Innenminister des Landes Nordrhein-Westfalen (2000), Section 1, §§ 2, 4, 5

[63]   cf. World Wide Web Consortium (2008), e.g. 2., 11., 21., 42.

[64]   cf. World Wide Web Consortium (2008), Abstract

[65]   cf. World Wide Web Consortium (2008), 1.

such as TouchKit for Vaadin. Although they can be overridden, a consistent theme is provided. Wessels, Purvis, Rahman (2011) support the importance of a consistent appearance, cross platform and for both the mobile as well as the desktop application, if existent[66]. Lica (2010) limits this to specific elements and points out, that mobile apps should provide just enough functionality to be useful and should not replicate the desktop optimized website.[67]

Other best practices such as utilizing a navigation bar at the page's top[68] for the main navigation are already adapted by different platforms and frameworks, such as Vaadin with TouchKit[69], iOS[70] and Android[71].

Best practices which are mainly determined by implementations of the developer, such as the usage of colors[72] or the chosen input methods[73] are often supported by the different platforms or frameworks but cannot be guaranteed by those. Even if different input methods such as a number pad for numeric inputs are provided by the framework or platform they still need to be adapted and utilized by the developer to act in line with the best practices.

The World Wide Web Consortium (2008) furthermore specifies a "Default Delivery Context"[74], which defines the minimal capabilities for mobile devices which should be supported. Tab. 4-1 illustrates the minimal capabilities suggested by W3C.

Nowadays it will be hard to match all of the requirements. E.g. a total maximum page weight of 20 kilobytes corresponds to the average file size of a 200 by 120 pixel JPEG-

---

[66]   cf. Wessels, Purvis, Rahman (2011), p. 2

[67]   cf. Lica (2010), p. 66

[68]   cf. World Wide Web Consortium (2008), 8.

[69]   cf. https://vaadin.com/book/-/page/mobile.components.html, last visited on 09/04/2013

[70]   cf.https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/ Navigation.html#//apple_ref/doc/uid/TP40006556-CH53-SW1, last visited on 09/04/2013

[71]   cf. http://developer.android.com/design/get-started/ui-overview.html, last visited on 09/04/2013

[72]   cf. World Wide Web Consortium (2008), 26., 27

[73]   cf.World Wide Web Consortium (2008), 55., 56., 57.

[74]   cf. World Wide Web Consortium (2008), 3.7 Default Delivery Context

Parameter	Value
Usable Screen Width	120px
Markup Language Support	XHTML Basic 1.1 delivered with content type application/xhtml+xml.
Character Encoding	UTF-8
Image Format Support	JPEG.
	GIF 89a.
Maximum Total Page Weight	20 kilobytes.
Colors	256 Colors, minimum.
Style Sheet Support	CSS Level 1. In addition, CSS Level 2 @media rule together with the handheld and all media types.
HTTP	HTTP/1.0 or more recent.
Script	No support for client side scripting.

Tab. 4-1: Default Delivery Context for Mobile Optimized Applications[77]

compressed file is about 10 kilobytes[75] and two images would already exceed the maximum page weight. With mobile devices such as a Samsung Galaxy S3, which has a minimum of 720 pixel wide display, 120 pixels are far too less, even as a minimum width. Also nearly any mobile browser supports client side scripting (e.g. JavaScript). For more detail, http://caniuse.com holds compatibility charts of different browser features for nearly any browser available. The parsing of JavaScript Object Notation (JSON) for example is supported by 93.41% of all mobile browsers according to caniuse[76].

Nevertheless, minimizing the total page size is still a concern. Wessels, Purvis, Rahman (2011) points out, that smaller pages lead to faster load times and therefore provide a more efficient experience for the user.[78] Nicolaou (2013) suggests different approaches to reduce page size as well as load time: Scripts and markup should be minified[79] and

---

[75]   Tested with 60% compression rate and a random photograph

[76]   cf. http://caniuse.com/#cats=JS_API, JSON parsing, last visited on 09/04/2013

[77]   cf. World Wide Web Consortium (2008), 3.7 Default Delivery Context

[78]   cf. Wessels, Purvis, Rahman (2011), p. 1066

[79]   cf. Nicolaou (2013), p. 49

included inline[80] where it is possible. Preloading components and reducing DNS lookups can also result in a faster user experience.[81]

Generally, Nicolaou (2013) recommend using a Content Delivery System (CDN), placing style sheets at the top of the source code, scripts at the bottom and using resized images rather than scaling them via HTML or CSS[82].

A study by Dahanayake et al. (2010) came to the result, that 71% of all responding web developers knew about the best practices, but only 11% said, that they would understand these, 56% have a vague understanding and 33% do not understand the best practices.[83]

Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009) adjusted and combined four different guidelines for application development, namely Shneiderman's Golden Rules of Interface Design, Seven Usability Guideline for Mobile Device (Abid Warsi, 2007), Human-Centred Design (ISO Standard 13407) and Mobile Web Best Practices 1.0 (W3C). Out of these guidelines, they developed the Three Layers Design Guideline for Mobile Application[84]. This guideline consists of three phases, which themselves represent different contexts, namely analysis (and the context of use), design (the context of medium) and testing (the context of evaluation). Tab. 4-2 illustrates this guideline.

This thesis will follow the Three Layers Design Guideline, as it is the latest guideline and includes parameters set in a multitude of other guidelines. The third phase will be shortened due to the temporal restrictions of this thesis. The exact process will be outlined in the following subsections.

---

[80] cf. Nicolaou (2013), p. 50

[81] cf. Nicolaou (2013), pp. 48, 49

[82] cf. Nicolaou (2013), pp. 49, 50

[83] cf. Dahanayake et al. (2010), p. 85

[84] cf. Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009), p. 430

[85] cf. Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009), p. 430, Table IV

	Phase	Context of Use and Activities
1	Analysis	**Use**: Specify user and organizational requirements
		1. Identify and document user's tasks 2. Identify and document organizational environment 3. Define the use of the system
2	Design	**Medium**: Produce design solution
		1. Enable frequent users to use shortcuts 2. Offer informative feedback 3. Consistency 4. Reversal of actions 5. Error prevention and simple error handling 6. Reduce short-term memory load 7. Design for multiple and dynamic contexts 8. Design for small devices 9. Design for speed and recovery 10. Design for "top-down" interaction 11. Allow for personalization 12. Don't repeat the navigation on every page 13. Clearly distinguish selected items
3	Testing	**Evaluation**: Evaluate design against user requirements
		1. Quick approach 2. Usability testing 3. Field studies 4. Predictive evaluation

Tab. 4-2: Three Layers Design Guideline for Mobile Application[85]

4.1.3 Internal Requirements

While developing a mobile frontend for ePill, it was important to us, that the main functionality of the web client will be optimized but not reduced in its main functionality. Therefore a good user interface is indispensable. All functionality should be accessible easily and without confusion for the user. Interactive elements such as buttons should be visible salient and have an immediate response to reduce the user's uncertainty.[86] The

---

[86]   cf. Norman (2002), p. 99

general design, the color scheme and the fonts should be used in line with the web application to improve the recognition value.

Another top priority is the accessibility of the app for as many users as possible. Therefore it is necessary to provide a cross-platform app that is accessible by as many mobile platforms as possible and to have an intuitive user interface which also enables elderly people to use it efficiently.

Modularity and flexibility is another important factor. ePill is designed to be flexible and scalable and the mobile client should incorporate the same idea. A scanning of barcodes on the packaging of pharmaceuticals could be implemented on a later stage to even further ease the use and increase the effectiveness.

4.2   Analysis

4.2.1   Assignment of a mHealth App Category

The mobile app does not differ from the web application in terms of privacy risks, content or connectivity. The mobile app aims to provide the same main functionality as the web application optimized for mobile devices, therefore it also belongs to the same connectivity category, the medical connectivity, as the web application. Also no data is stored on the device and no additional usage data is sent to the server.

We plan to implement a SSL-encrypted connection to the server as soon as the later mentioned is capable of accepting and responding via SSL-encryption.

We would suggest to categorize the ePill mobile application as a low privacy risk, drug- and safety-related medical connectivity mHealth application and it should be categorized the same as the web application.

Possible future features might change the classification (e.g. the addressed barcode scanning) if data handling or storage is altered and with it other privacy risks may arise.

4.2.2   The Different Operation Systems

4.2.2.1   Android

Android is a mobile OS developed by the Open Handset Alliance[87], with Google being one of the biggest members. It is Linux based and was unveiled in 2007. Android is released by Google under the Apache License and is therefore Open Source.[88] Developing for Android requires the Android SDK (or NDK) to be installed on the development computer. Developing apps with the SDK is done by writing Java code and defining the layout in specific XML[89]. Android apps are executed in the Dalvik managed runtime[90] by default, except if they utilize the NDK. With the NDK apps can be (partly) written in C or C++ and are executed outside the Dalvik runtime which can be helpful for reusing existing code or executing CPU-intensive operations.[91]

While Android is adapted by many manufacturers as well as consumers, a software-based and a hardware-based fragmentation is visible.[92] Fragmentation means in this case a variety of different software and hardware which is at least partly incompatible. This fragmentation offers the user the choice to find exactly what he is looking for and enables more personalization. However, fragmentation may lead to non-consistent applications on different devices as well as delays in updates. According to Google, an Android version released 2010 (2.3 "Gingerbread") still has a distribution of around 30.7%.[93] This also implies that developers do not only have to regard latest versions of the OS, but also older versions. This means in return, that developers cannot always take full advantages of new capabilities as well as they need to pay much more attention to backwards compatibility.

For Android multiple IDEs are available. Eclipse[94] is a popular IDE and is officially supported by the Android SDK.[95] Android Studio, an Android optimized version of In-

---

[87] http://www.openhandsetalliance.com, last visited on 09/02/2013

[88] cf. http://source.android.com/source/licenses.html, last visited on 09/02/2013

[89] cf. for further details http://developer.android.com, last visited on 06/09/2013

[90] cf. http://source.android.com/devices/tech/dalvik/index.html, last visited on 09/09/2013

[91] cf. http://developer.android.com/tools/sdk/ndk/index.html, last visited on 09/09/2013

[92] cf. for this and the next three following sentence Dan Han et al. (2012), pp. 83, 92

[93] cf. http://developer.android.com/about/dashboards/index.html, last visited on 09/09/2013

[94] http://www.eclipse.org, last visited on 09/09/2013

[95] cf. http://developer.android.com/sdk/installing/bundle.html, last visited on 09/22/2013

telliJ IDEA[96], is still in development but already a stable IDE and greatly supported by Google[97].

Apps can be distributed directly to single phones or via an app store, e.g. the Google Play Store[98]. The Google Play Store has guidelines, which must be followed[99], but no review process, in which compliance with the guidelines is verified, is performed.[100]

4.2.2.2 iOS

iOS is the proprietary OS developed by Apple for mobile devices. It was first introduced in 2008. In contrast to the Android OS only Apple develops hardware and software that runs the iOS operation system. According to Apple, 94% of all active iOS devices run the latest version of iOS 6.[101] Compared to Android, all versions of iOS released before 2011 have a cumulative distribution of only 1%. Hardware-based fragmentation on iOS is mainly based on the screen size: Two different for the phones and one for the tablets.

iOS apps can only be developed in Xcode[102], which is only available for Apple Mac. Xcode combines user interface design and coding. Coding is mainly done by writing Objective-C code but also supports native code written in C or C++. Designing interfaces is done with an Interface Builder called user interface integrated into Xcode.[103]

In contrast to Android, iOS apps can only be published via the Apple App Store[104], or

---

[96]   http://www.jetbrains.com/idea/, last visited on 09/09/2013

[97]   cf. http://developer.android.com/sdk/installing/studio.html, last visited 09/09/2013

[98]   https://play.google.com/store, last visited on 09/09/2013

[99]   https://play.google.com/about/developer-content-policy.html, last visited on 09/09/2013

[100] for further information see http://developer.android.com/distribute/googleplay/publish/preparing. html, last visited on 09/24/2013

[101] cf. for this and the first following sentence https://developer.apple.com/devcenter/ios/checklist/, last visited on 09/11/2013

[102] https://developer.apple.com/xcode/, last visited on 09/09/2013

[103] cf.   https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_ Overview/Edit_User_Interfaces/edit_user_interface.html#//apple_ref/doc/uid/TP40010215-CH6- SW1, last visited on 09/09/2013

[104] http://appstore.com, last visited on 09/11/2013

on registered devices with a special license by Apple[105]. For submitting apps to the Apple App Store, one must obtain a developer license by Apple[106]. After submitting, all apps are reviewed and compared to Apple's guidelines[107].

4.2.2.3 Windows Phone 7 and 8

Windows Phone 7 was released in 2010 as the successor of Windows Mobile. Windows Phone 8, the phone version of Windows 8, were released in 2012. Both utilize the "Metro" design, a tile-based design.

Development for Windows Phone requires the Visual Studio IDE.[108] Visual Studio is only available for Windows. C# or Visual Basic are the main programming languages and XAML is used for user interface design[109]. C++ can be utilized for graphic intensive applications.

The Windows Phone Store[110] is a closed store like the Apple App Store. This means, that an enrollment is needed to publish apps and the release of an app is preceded by a review process.[111] Unlike the other stores, the Windows Phone Store offers the possibility to try apps out with reduced functionality.

4.2.2.4 other

Depending on the source of statistics, the lead market share is held by different OS. Nevertheless other OS, e.g. Symbian, which was an important OS in 2008 with 47% market share of smartphone OS[112], is nowadays not listed at all or with less than 10% market

---

[105]   cf. https://developer.apple.com/programs/ios/enterprise/, last visited on 09/11/2013

[106]   cf. https://developer.apple.com/programs/ios/, last visited on 09/11/2013

[107]   https://developer.apple.com/appstore/guidelines.html, last visited on 09/11/2013

[108]   http://www.microsoft.com/visualstudio/, last visited on 09/11/2013

[109]   cf. for this and the first following sentence http://msdn.microsoft.com/en-US/library/windowsphone/develop/ff402529(v=vs.105).aspx, last visited on 09/11/2013

[110]   http://www.windowsphone.com/store, last visited on 09/11/2013

[111]   http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184843(v=vs.105).aspx, last visited on 09/11/2013

[112]   cf. "Canalys research release 2008/112" cited by Lin, Ye (2009), p. 617, Figure 1

share[113, 114, 115].

We will not take these OS into account, whose combined marketshare is around only 10%, as this would exceed the time frame allocated for this thesis.

4.2.3 Possible Frameworks and Technologies

4.2.3.1 Completely native

Building native apps for supporting Android, iOS and Windows Phone, means maintaining three different projects in three different programming languages and three different user interface definitions. But native apps offer the most seamless user interface integration into the OS and the best performance. Native apps are developed for a specific OS and utilize the user interface controls provided by the OS.

While the seamless user interface integration could help people using the app in a familiar context, performance is not an issue for the ePill project. The costs for learning the specific aspects of those frameworks, developing and maintaining the three implementations are not reasonable for the given time frame of this thesis.

Additionally, we do not have an existing web service which the app could utilize to receive data from the server. With the logic already implemented, the currently used framework Vaadin does not offer a way to provide a web service. This additional effort is definitely a decisive argument.

4.2.3.2 HTML 5, jQuery mobile and Phone Gap

Providing an app for nearly any mobile device is possible with a web app. Based on web technologies like HTML, CSS and JavaScript, apps can be brought to nearly any mobile OS with only one implementation.

As HTML5[116] and JavaScript is supported by most mobil devices, frameworks like jQuery

---

[113]   http://gs.statcounter.com/#mobile_os-ww-yearly-2008-2013, last visited on 09/11/2013

[114]   http://www.idc.com/getdoc.jsp?containerId=prUS24257413, last visited on 09/11/2013

[115]   http://blogs.strategyanalytics.com/WSS/post/2013/08/01/Strategy-Analytics-Android-Captures-Record-80-Percent-Share-of-Global-Smartphone-Shipments-in-Q2-2013.aspx, Exhibit 1, last visited on 09/11/2013

[116]   http://www.w3.org/TR/2012/WD-html51-20121217/, last visited on 09/11/2013

mobile[117] provide a common looking user interface without much additional effort. Phone-Gap[118] enables web-based apps to access the device's capabilities like the camera or local storage. A detailed overview as well as unsupported features or OS can be found on PhoneGap's feature page[119].

Still, with this approach we would need a web service, which would be consumed with jQuery. The web app approach lessens the effort by developing multiple apps with only one code basis and user interface definition but still has the time consuming need for a new web service.

4.2.3.3 Xamarin

Xamarin[120], also known as MonoTouch, is an IDE and framework, which produces native apps for Android, iOS and Windows Phone from just a single C# code basis. Different user interface definitions are still needed, but it combines the advantages of native apps and web apps.
Xamarin used with Visual Studio or Xamarin Studio, while Visual Studio is only available for Windows, Xamarin Studio is available for both Windows and Mac OS X.

Also apps developed with Xamarin would need an additional web service, which cannot be provided in the given time. Additionally, Xamarin is only free by meeting specific requirements, so it is likely to be more expensive than providing a web app.

4.2.3.4 Vaadin and TouchKit

Vaadin[121] is a Java framework, with which the developer only needs to write application code and define a user interface included in the Java code. The framework builds the user interface as HTML and handles all communication between the client and the server.

---

[117] http://jquerymobile.com, last visited on 09/11/2013

[118] http://phonegap.com/, last visited on 09/21/2013

[119] http://phonegap.com/about/feature/, last visited on 09/11/2013

[120] http://xamarin.com, last visited on 09/21/2013

[121] https://vaadin.com, last visited on 09/11/2013

TouchKit[122] enables a Vaadin project to easily add a mobile web client. It provides various user interface controls which were optimized for mobile devices. It also provides access to the device's capabilities, such as positioning, offline storage or camera.

TouchKit supports iOS 5 or newer, Android 2.3 or newer and Windows Phone 8. Vaadin as well as it's add-on TouchKit, is focused on WebKit[123]-based browsers[124], although it is compatible to most of the mobile browsers. Nevertheless, browsers which are not based on WebKit might be completely unusable or might have layout defects.

There are advantages to using Vaadin and TouchKit for ePill. More specifically, an additional web service is not necessary when using the existing code and simply adding a TouchKit-based user interface which seamlessly adapts to the existing code.

4.2.4   The Choice for Vaadin and TouchKit

Finally we chose Vaadin and the TouchKit add-on as the framework for the mobile frontend. The main reason to this is the lack of an accessible web service in Vaadin itself. Without a web service, we would have had to build a web service first to access the data layer or the application's logic.

Furthermore we wanted the complete system to be as homogenous as possible. Utilizing the TouchKit Add-On for Vaadin, we reused as much existing code as possible and infrastructure by only adding another layer. This results in a much improved maintainability as the coding style is the same for the web application and the mobile application. Additionally no additional IDEs or frameworks need to be included or maintained.
The missing compatibility to not WebKit-based browsers as well as the abstracted user interface definition might be drawbacks. With a less abstract user interface definition we would have more options to fine tune the resulting user interface.

---

[122]   https://vaadin.com/touchkit, last visited on 09/11/2013

[123]   http://webkit.org, last visited on 09/11/2013

[124]   cf. https://vaadin.com/book/-/page/mobile.considerations.html, last visited on 09/11/2013

4.3   The Planning Process

As already mentioned, the ePill system is already a functional prototype and the yet to develop mobile frontend is mostly an additional user-interface. Therefore no system-wide planning, including e.g. data organization, is needed but the data handling and user interfaces have to be planned.

Having the already existing application logic in mind, the planning of data handling is reduced to a functional planning[125]. To be able to have the functions available for the user, we have to combine the planning of the functional aspects with the user interface, therefore we will plan the user's flow through the application (user flow) which pays attention to the accessibility of the of the functions of the mobile web app in the right context during the use of the app, and combines functional and user interface planning. For the ease of explanation, we added some mockups and screenshots. We decided to show them uncolored to prevent distraction from highlighting colors and because the color scheme is not final yet. We decided to focus on smaller mobile devices such as phones and not on tablets, because smaller user interfaces can be expanded and are still usable. If the user interface is designed for a bigger screen and is presented on a smaller screen, either the elements become too small or the user needs to scroll through the interface. We will optimize the app's implementation for various layouts depending on the device's screen size at a later stage.

The three major functions of ePill (Search, Display and Supplementing Services) should be accessible as fast as possible. The display functionality and some supplementing services (e.g. the term explanation functionality) are only able to present results if a pharmaceutical is selected. So we designed Fig. 4-1 as a minimal starting screen with every main function quickly accessible. Due to some missing user interface controls provided by Vaadin TouchKit, such as the search bar, we finally implemented the start screen as illustrated in Fig. 4-2. This furthermore resulted in a more separated presentation of search string input view and result view, as presented in Fig. 4-3 and Fig. 4-4.

---

[125]   planning of the different functions available to the user and how they are accessed with which results and how error handling is performed
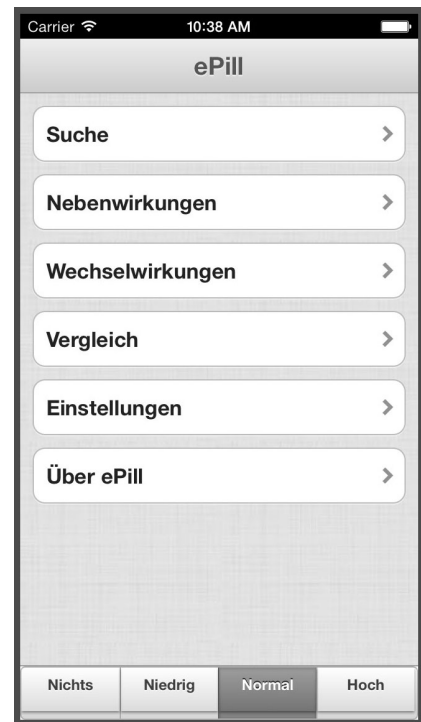
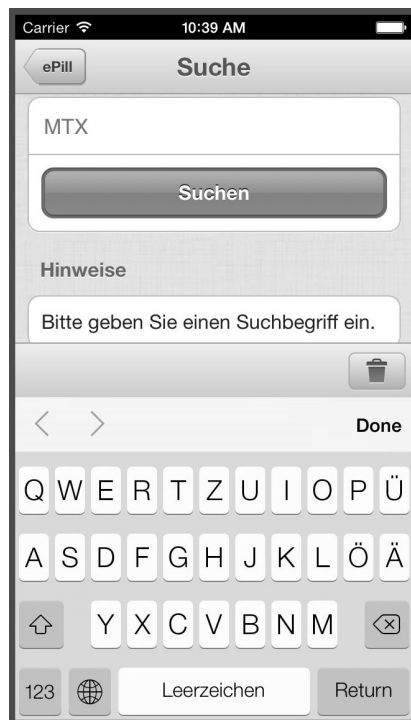Fig. 4-1: Main Screen Mockup

Fig. 4-2: Final Main Screen

Fig. 4-3: Search Input Screen

Fig. 4-4: Search Result Screen

Throughout the planning and designing we tried to reproduce common patterns and user interface controls, that are well know through the usage of other apps, and therefore we utilized the navigation back button on the top left corner, which is illustrated by a leftwards oriented arrow-shaped button. Central navigation which specifies the view is centered in the main content area at the center of the screen.

Most of the views offer a toolbar at the bottom of the screen. This toolbar contains controls which are used for navigation inside the view at the center of the screen, such as the ability to go back, to the top or to navigate to a specific headline. In the pharmaceutical details view with all the information visible, it can be very handy to jump right to the header one is interested in. Additionally, some controls are added in the toolbar which enable further interaction with the information displayed, e.g. adding the currently visible pharmaceutical to the comparison list, which we will discuss soon.

The web application has rich customization possibilities. One can adjust the font size, the details of pharmaceuticals to be displayed and the layout of the web application itself. As it turned out, it is not applicable for the user to change the mobile apps layout, because the user interface so too compact to add additional elements and too few elements are visible to deem it necessary to hide any given element.

The comparison list is a concept added to the mobile application and derived from an element in the web application. The web application provides a view in which pharmaceuticals can be added and afterwards, via a button click, be compared. Aggregated information such as adverse reactions can also be listed. Because we have much less available screen space in the mobile app, we decided to keep the list globally available throughout the entire app. For example, if one searched for a pharmaceutical and has selected one to display detailed information, he can easily add this pharmaceutical (via a button in the toolbar) to his comparison list. If one now selects the "side effects" functionality on the main screen, he sees his comparison list and can add addition pharmaceuticals to it. This screen is illustrated in Fig. 4-7. One is now presented a search input screen but can in the search result screen only tick on those pharmaceuticals he wants to add to his comparison

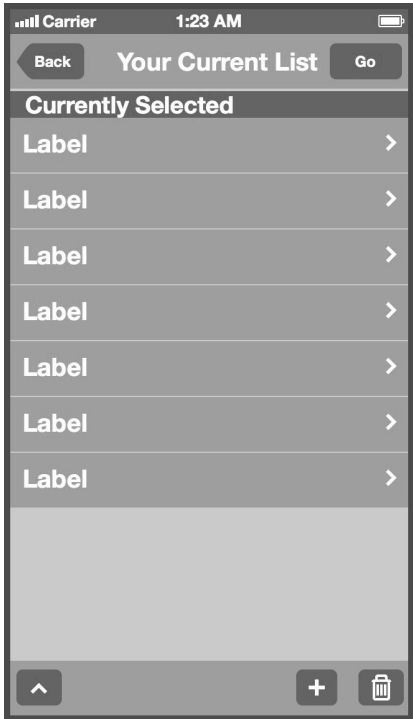list, which is illustrated in Fig. 4-8.
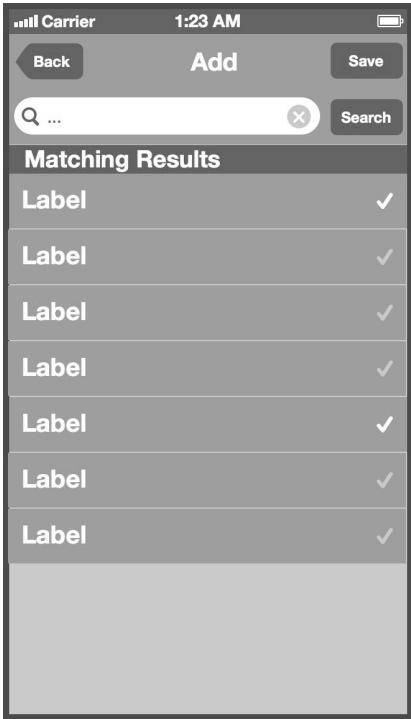

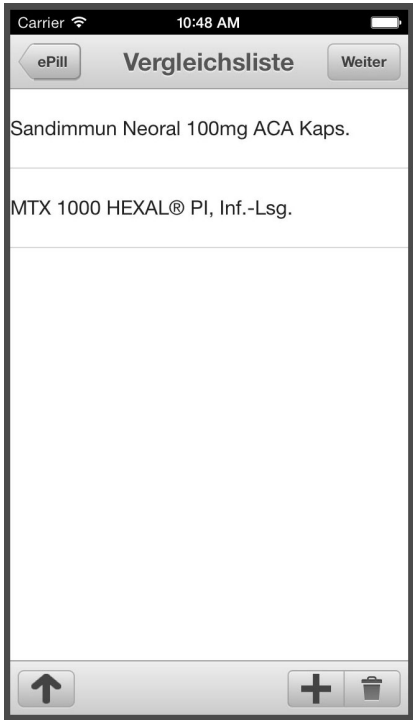Fig. 4-5: Comparison List Mockup

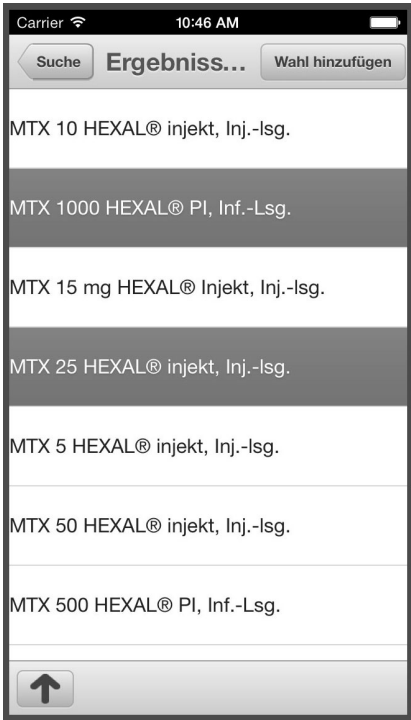Fig. 4-6: Results Screen Mockup

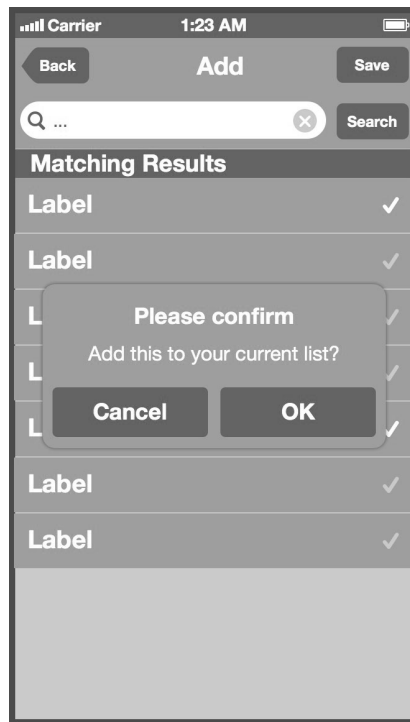Fig. 4-7: Comparison List Screen

Fig. 4-8: Results Screen
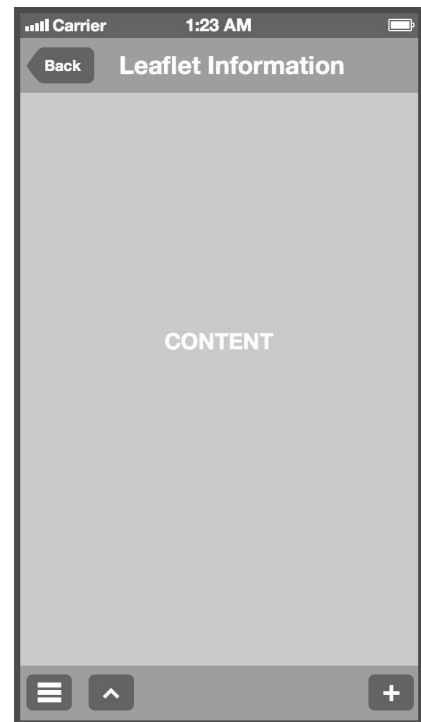
Fig. 4-9: Confirm Dialog Mockup
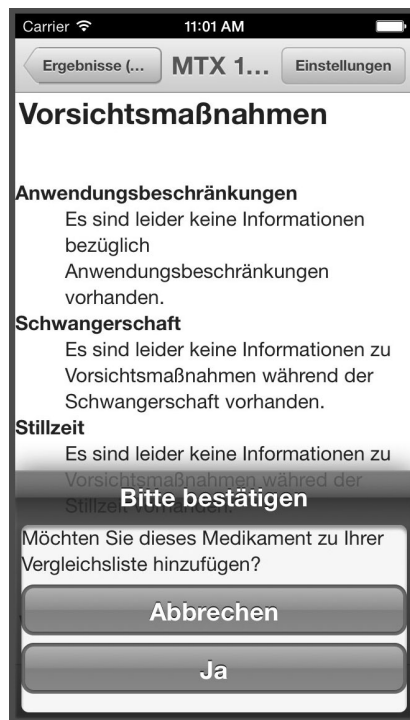
Fig. 4-10: Details Screen Mockup
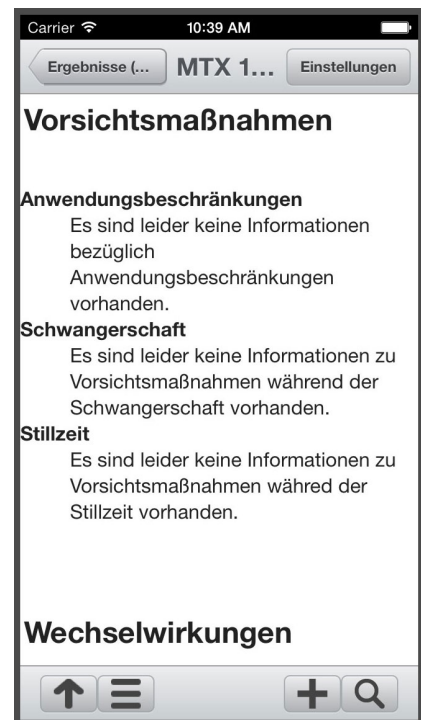
Fig. 4-11: Confirm Dialog

Fig. 4-12: Details Screen

Fig. 4-7 also highlights the toolbar. The leftmost button scrolls back to the top, whereas on the right side we have an add button which brings up the already discussed search view and the trash button, which clears the comparison list.

The button on the top right corner, labeled "Go", starts the currently selected function, e.g. adverse reactions on the comparison list.

To prevent users from misusage, we will add dialogs asking to confirm the chosen operation where adequate. This currently only effects the delete and add operations of the comparison list, as illustrated in Fig. 4-11.

Fig. 4-12 illustrates the pharmaceuticals details screen. This screen presents all information for a specific pharmaceutical, e.g. after the selection of one pharmaceutical in the search results screen or from the comparison list as illustrated in Fig. 4-7. Basically, this screen consists of the back navigation button in the top left corner, the toolbar in the bottom and the centered main content area as most of the mobile app's screens. While the main content area displays the chosen leaflet information in a segmented way, the toolbar offers different operations. On the lower right corner, the add button allows the user to add the currently selected pharmaceutical to his comparison list. The left side of the toolbar offers a back to top button just such as the comparison list screen as well as a navigation button, enabling the user to jump to any of the sections by which the main content is segmented. A click on this button reveals a list of the sections through which the user can scroll and select one to navigate to. Another click on this button closes the list as well as the section selection.

4.4 The Implementation Process

We had some issues at the beginning of the implementation of the planned user flow, because Vaadin and TouchKit differ to other frameworks we worked with beforehand. For instance, TouchKit does not provide a search bar which can be attached to the navigation bar as planned in the mockups, furthermore the ePill web application is currently written in Vaadin 6, whereas TouchKit 3 is based on Vaadin 7. We tried to utilize Vaadin 6 and TouchKit 2 for the mobile app but as it turned out, TouchKit 2 has some drawbacks in comparison to TouchKit 3, such as a smaller documentation, tutorials and example projects. The choice for Vaadin 7 and TouchKit 3 resulted in, that we could not just include the ePill web application and directly access the business logic, as adjustments were

needed to upgrade a Vaadin 6 project.

We decided to include as much as possible from the web application and copy and modify as little as possible. During further development, after the web application was upgraded to Vaadin 7, both projects could be merged together and code redundancies can be removed. The choice for Vaadin 7 brought additional work as we had to adjust inner code references to e.g. the main window and application because their handling did change.

Programming with Vaadin is easy after the first steps into the framework were successful. The coding style supposed by the framework was consistent and helped achieving fast results. Nevertheless, missing controls are a drawback and the inflexibility of the framework is both an advantage and disadvantage. While the given user interface controls and the limited influence on their exact appearance help providing a consistent user interface, they prevent from optimizing and fine tuning. For instance, it is not easily possible to access device specific functionality by utilizing APIs of the specific OS. PhoneGap can be integrated but this also means addition effort for utilizing an additional framework.

The lack of components such as alert- and confirm-dialogs or search bars is compensable. During the development, we utilized popovers to imitate the well known appearance of dialogs in most of the mobile OS. Vaadins popover component is functional, as it provides an overlay with a responsive layout[126] for different screen sizes but is not completely customizable. For instance, the bar at the popovers top cannot be customized and buttons cannot be added to this bar. Vaadins popover has also a much bigger border than we would have used with a common dialog component. This reduces the useable content area. Additionally the borders distract the user and reduces the focus on the content and buttons.

We implemented a workaround for the missing search bar attached to the navigation bar by adding an additional view. As stated beforehand, we planed to attach the search bar directly beneath the navigation bar to enable the user to quickly access the search functionality. Results were planned to be displayed as an overlay list. We had to abandon

---

[126]   a layout which adjusts to the screen size

the idea of an overlay list as well. Overlays looked either unfamiliar or complicated the use of the app, especially for dismissing the results. Finally we chose to display a view containing an input field, a search button and a label displaying information such as the minimum length of the query. The results are displayed in another standalone view as a plain list.

Another issue we discovered was the drop for dynamic style sheets and themes in Vaadin 7. In Vaadin 6 one could swap the used theme during runtime, which would be useful to adjust the font size. In the allocated time frame, we were not able to implement a workaround. We tried adjusting the font size dynamically by adding CSS classes to the different user interface controls but this either resulted in a high complexity for the algorithm to discover all matching controls nor high memory usage if we were to keep references for the controls.

Thanks to the automatic handling of references between the client- and the server-side, E.g. the settings screen for customizing the displayed leaflet information could be implemented quickly. Nevertheless it was not possible to receive specific events such as a "will appear" or "did appear" for views, which are called either if the user navigates to this screen or is navigated backwards to this screen. These events are needed for refreshing the view or specific controls if needed. Java and Vaadin offer listeners for "value changed" events but those still need a reference to the controls, which need to be passed along through the views. The main screen contains a toolbar at the screens bottom to select a preset of visible information on leaflets. If the user decides to change the preset in a lower view, this toolbar needs to be updated as well. Additionally to the increased reference count this method has the drawback that the toolbar gets updated on every value change, although it is not visible. With a "will appear" event the update would have been done only once immediately before the toolbar becomes visible.

Parts of the layout from the web application could be reused with only minor adjustments, thanks to Vaadins seamless TouchKit integration. We had to make adjustments on the layout of tables, especially the count of columns as well as the preloaded content. We reduced the displayed information to a minimum for usability but comprehensibility.

Comparing two pharmaceuticals can only be effective, if related information is presented closely. During the development we experimented with a vertical instead a horizontal layout, meaning that each related information of the two pharmaceuticals are presented beneath each other, not side by side, but it became very confusing because of the doubled page length. With the headlines beneath each other, one needed to scroll too much to compare the pharmaceuticals directly. We concluded that a horizontal layout and a landscape orientation will be best in this case. The landscape orientation provided enough screen space on most devices to have associated headlines side by side which eased the comparison.

4.5   Validation of the Mobile App

In this section we will compare the developed app to the norms and best practices highlighted in section 4.1 Preconditions, as well as to all requirements stated in section 4.2 Analysis.

Because no user-related data is stored, neither personal nor usage related, no conflicts with the norms mentioned in section 4.1.1 Norms for Mobile Apps were discovered.

Vaadin and TouchKit support developers by providing a default theme for applications to maintain a consistent user interface, design and color wise. Despite the statement of Wessels, Purvis, Rahman (2011) that a consistency to a desktop application should be maintained[127], we focused on an easy to use user interface on mobile devices. Some parts look very familiar on both applications, e.g. the detailed informations or the comparison of pharmaceuticals, but the navigation is absolutely different as a result of the limited screen size and a touch- instead of mouse-based operation. This differentiation was chosen with the statement of Lica (2010) in mind that the mobile app should only provide enough functionality to be useful[128]. This resulted likewise in a missing customizable tab-bar or different sidebars. These elements might be operational on a bigger screened device such as tablets, but it is not useable on most mobile phones. Further development

---

[127]   cf. Wessels, Purvis, Rahman (2011), p. 1067

[128]   cf. Lica (2010), p. 66

of the mobile app might result in an optimized tablet user interface and integrate this functionality.

In sections 4.3 The Planning Process and 4.4 The Implementation Process we stated that we utilized a navigation bar and a toolbar as main interaction controls for navigation and direct interaction such as clearing lists. These two concepts of placing controls on the edges of the screen are also recommended by World Wide Web Consortium (2008).[129]

We also tried to follow common input methods. As no specific input fields for e.g. numbers, were needed, the only specialization we could use was the enter-key equivalent on mobile devices, often a button on the lower right corner of the simulated on-screen keyboard.

The World Wide Web Consortium (2008) suggested a "Default Delivery Context"[130] illustrated in Tab. 4-1 Mobile Default Delivery Context for Mobile Optimized Applications, but Vaadin does not leave much to the developer to optimize the rendering of the controls as well as the organization of requests and client-side handling of data. Developers can only optimize used images and CSS style sheets. Because only very few images are used and not much custom styling is applied, we could not pursue this recommendation in particular.

Comparing this app to the Three Layers Design Guideline developed by Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009), illustrated in Tab. 4-2, results in the following: While phase one, the analysis, was already done by Dehling, Sunyaev (2012a), phase two was one of the purposes of this thesis.[131] We tried to implement shortcuts where possible with the help of toolbars and quick access to settings, specially in the detail view, and offered informative feedback where possible, e.g. for the search right beneath the input field or in popovers for detailed information. Using the same theme and general user interface layout led to the proposed consistency as well as the optimization for small devices. The utilization of a navigation bar at the top also compiles with the proposed "top-down" interaction or navigation. Vaadin as the framework as well as the already

---

[129]   cf. World Wide Web Consortium (2008), 8.

[130]   cf. World Health Organization (2011), 3.7 Default Delivery Context

[131]   cf. for this and the following paragraph Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009), p.430, Table IV

existing code support error prevention and handling through catching thrown exceptions and giving a direct response to the user. Personalization is made possible with the settings view which enables the user to select only those information to be displayed he is interested in. Norman (2002) states, that grouping controls into logical and functional modules as well as hiding infrequently used controls is a good way to reduce complexity.[132] We grouped the main functions as buttons onto the main screen, added a toolbar at the bottom for quick access to the detail level settings and hid all functionality on another screen, accessible via the mentioned buttons.

Phase three, testing, could only be done superficially until the end of this thesis' time frame. We did only a short functional testing. Further development might pursue a more detailed testing of the mobile app.

The internal requirements were compiled with, buttons were made salient and interaction with them resulted in immediate feedback. For instance, the search button disables itself until the server processed the request hence the user cannot execute the search twice. The mobile app is available on nearly any mobile platform because of the good distribution of HTML 5 compatible browsers. Also, the app is as modular as the web application because much code was reused and only adapted to the changes in Vaadin 7 compared to Vaadin 6 and the layout was optimized for small screens. The color scheme used by the web application was not adapted because of the more general familiarity among mobile OS users with the TouchKit default theme because of its similarity to the appearance of iOS apps as well as other mobile web apps.

We propose that further development might improve the utilized default TouchKit theme, for a more effective and intuitive use of the mobile app. We would set the focus on the popovers which are currently not matching perfectly to the overall appearance. Nevertheless, the mobile app follows the guidelines stated in section 4.1 Preconditions and should provide a good user interface and functionality. A dedicated usability test may reveal some remaining weak points.

---

[132]   cf. Norman (2002), p. 175, 6.9

5. Lessons Learned

mHealth apps do have some similar requirements as any app, but have to deal with much stronger information security concerns compared to casual apps. They also need to pay more attention to usability and availability than most other apps by definition.[133]

In the case of ePill information security is not an issue because no user related information is stored. But developing with Vaadin was not always a good choice. Vaadin simplified the development process but slowed it down at the very beginning. The app is not as perfectly designed as it could have been with more influence on the final layout on the client-side, especially on the user interface in terms of available user interface controls.

After having read the Vaadins Beginner Guide[134] we only had minor issues understanding Vaadins architecture and development went mostly quick and easy. As already stated, some missing controls were a drawback to us and workarounds had to be found, but all in all it is a good solution for a quick development without the need of further knowledge in HTML, CSS or JavaScript. Having not to deal with cross-browser-optimizations was a relieve.

After the development of the mobile app we would nevertheless propose a different approach. If Vaadin is already utilized in the existing system, it is good to reuse the code. But if not, or if a web service is available, we would suggest developing native applications, maybe with Xamarin[135]. This offers the possibility to fine tune the user interface much more, offer a much more familiar look on the different OS and much more user interface controls are available. While the web app developed with Vaadin looks on all OS similar and similar to iOS, with native apps the different apps would incorporate themselves more into the environment of the OS. It is worth the additional effort of developing a standalone web service to provide the data for a mobile client and different user interfaces, at least their OS-specific definition for the better accessibility. Additionally, OS like iOS offer native apps the possibility to run in background and perform specific tasks

---

[133]   cf World Health Organization (2011) cited by Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013), p. 2

[134]   https://vaadin.com/tutorial, last visited on 09/21/2013

[135]   http://xamarin.com, last visited on 09/21/2013

energy efficiently, while web-based apps are forced to quit on exit. This constraint does not affect ePill but e.g. monitoring apps need a continuing execution.

Frameworks such as PhoneGap[136] provide accessibility to extended functionality of the device for most mobile OS but still do not offer background execution. For instance, Apple offers APIs only for native apps.[137] Android offers similar to Apple Services, the execution environment for long running tasks, only to native apps.[138]

Using native apps also eases the integration of accessibility features such as font enlargement or voice guided navigation inside the app. Most modern OS provide many accessibility features which can not or only partially be utilized inside a web-based app. A voice-guided navigation could be useful for ePill for either impaired people or in situations in which one is not able to operate by touch.

We furthermore learned that planning plays an important role for a fast and successful implementation. For a successful planning, knowledge of the used software and their possibilities is important. During planning we had only superficial knowledge about Vaadin and TouchKit, we inspected sample code and user interfaces, which made us believe that most controls from native applications can be utilized as well. If we knew during development that many controls are not implemented already, our planned user interface would have looked different and we would have saved some time during the implementation.

---

[136] http://phonegap.com, last visited on 09/21/2013

[137] cf. https://developer.apple.com/library/ios/documentation/iphone/conceptual/ iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html, last visited on 09/24/2013

[138] cf. http://developer.android.com/guide/components/services.html, last visited 09/24/2013

6. Conclusion

Adding and rating the subgoals, we come to the conclusion that this thesis' goals are achieved and that we gave not only an overview about the current state of guidelines for mHealth apps, but also developed a functional prototype of a mobile frontend for the ePill web application. We stated all processes we passed through, as we aimed to do.

The literature review showed that not many guidelines or theoretical development frameworks specialized on mHealth apps or even for mobile apps in general are available. Some guidelines are in need for revision, due to the changes in mobile development since the release of the literature. The goal to provide a short overview over the current state of best practices and guidelines is thereby fulfilled.

We reported any issues and challenges we discovered during the development of a mobile frontend for the ePill system in sections 4.3 and 4.4 and summarized them in section 5. The goal for providing an overview over the lessons learned during the systematic development is thereby fulfilled.

A validation of the developed mobile frontend as required by the objectives of this thesis was performed in section 4.5. The validation confirmed the development of the mobile frontend in compliance with the stated guidelines and best practices. The goal for developing a mobile frontend and its validation is thereby fulfilled.

All modifications or deviations between the web application and the mobile app or between the planning and the implementation were discussed and explained, such as the altered search bar in sections 4.4 and 4.5. The developed mobile app contains all needed functionality and has a mobile optimized user interface which is available on nearly any modern mobile device. It is therefore in compliance with the aim to provide a cross platform mobile frontend for ePill.

A possible point of criticism is that there was no time for concentrating on fine tuning and deep testing of the mobile app. The missing knowledge of Vaadin and TouchKit resulted in a user interface planning which was not accurate. We had to find workarounds for the missing user interface controls we integrated in our planning. Additional knowl-

edge about Vaadin and TouchKit would have avoided this time consuming process. Those workarounds could be further improved by refining the CSS styling or implement additional user interface controls for Vaadin and TouchKit.

Additional literature for designing user interfaces in general could have been reviewed. For example "The Design of Everyday Things" by Norman (2002) could have been integrated and compared to the "Three Layers Design Guideline" for mobile application by Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009). These additional general design guidelines could have improved the user interface design.

Another point of criticism is that some of the discovered issues are specific to Vaadin and TouchKit, such as the missing user interface controls or the missing influence on the final user interface appearance. Other issues, such as the missing familiar look of the mobile app on Android or Windows Phone devices are applicable to any app which user interfaces are not defined independently for every OS.

Bibliography

Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Dahlan (2009)

Ayob, Nurul Zakiah binti, Hussin, Ab Razak Che, Halina Mohamed Dahlan: Three Layers Design Guideline for Mobile Application. In: Information Management and Engineering, International Conference on. 2009, pp. 427–431

Badashian et al. (2008)

Ali Sajedi Badashian, Mehregan Mahdavi, Amir Pourshirmohammadi, Minoo Mona-jjemi nejad: Fundamental Usability Guidelines for User Interface Design. In: Computational Sciences and Its Applications, 2008. ICCSA '08. International Conference on. 2008, pp. 106–113

Bundesregierung der Bundesrepublik Deutschland (1996)

Bundesregierung der Bundesrepublik Deutschland: Telekommunikationsgesetz: TKG, 1996. http://www.gesetze-im-internet.de/tkg_2004/, visited on 09/07/2013

Bundesregierung der Bundesrepublik Deutschland (2007)

Bundesregierung der Bundesrepublik Deutschland: Telemediengesetz: TMG, 2007. http://www.gesetze-im-internet.de/tmg/, visited on 09/07/2013

Dahanayake et al. (2010)

Ajantha Dahanayake, Caroline Collier, Daniel Glenzer, Tanya Goette, Richard Welke: Mobile Website Engineering and Mobile Web Best Practices Guidelines: A Reality Check. In: Journal of International Technology and Information Management. Nr. 2, Jg. 19, 2010, pp. 79–IV

Dan Han et al. (2012)

Dan Han, Chenlei Zhang, Xiaochao Fan, Abram Hindle, Kenny Wong, Eleni Stroulia: Understanding Android Fragmentation with Topic Analysis of Vendor-Specific Bugs. In: Reverse Engineering, Working Conference on. 2012, pp. 83–92

Dehling, Sunyaev (2012)

Tobias Dehling, Ali Sunyaev: Architecture and Design of a Patient-Friendly eHealth Web Application: Patient Information Leaflets and Supplementary Services. In: AMCIS 2012 Proceedings. 2012, pp. 1–8

Dehling, Sunyaev (2012)

Tobias Dehling, Ali Sunyaev: Improved Medication Compliance through Health IT: Design and Mixed Methods Evaluation of the Application ePill: Research-in-Progress. In: Thirty Fourth International Conference on Information Systems, Milan. 2012, pp. 1–11

Dehling, Sunyaev (2013)

Tobias Dehling, Ali Sunyaev: Information Security and Privacy Implications of mHealth Apps: An Overview. 2013, pp. 1–12

Der Innenminister des Landes Nordrhein-Westfalen (2000)

Der Innenminister des Landes Nordrhein-Westfalen: Gesetz zum Schutz personenbezogener Daten: DSG NRW, 2000. https://recht.nrw.de/lmi/owa/br_bes_text?anw_nr= 2&gld_nr=2&ugl_nr=20061&bes_id=4908&aufgehoben=N&menu=1&sg=0, visited on 09/08/2013

d'Heureuse et al. (2012)

Nico d'Heureuse, Felipe Huici, Mayutan Arumaithurai, Mohamed Ahmed, Konstantina Papagiannaki, Saverio Niccolini: What's app?: a wide-scale measurement study of smart phone markets. In: SIGMOBILE Mob. Comput. Commun. Rev. Nr. 2, Jg. 16, 2012, pp. 16–27

Future of Privacy Forum, Center for Democracy & Technology (2011)

Future of Privacy Forum, Center for Democracy & Technology: Best Practices for Mobile Application Developers: App Privacy Guidelines. In: Future of Privacy Forum and the Center for Democracy & Technology. 2011, pp. I–17

Hasan, Isaac (2011)

S.S. Hasan, R.K. Isaac: An integrated approach of MAS-CommonKADS, Model–View–Controller and web application optimization strategies for web-based expert system development. In: Expert Systems with Applications. Nr. 1, Jg. 38, 2011, pp. 417–428

Istepanian, Jovanov, Zhang (2004)

R.S.H. Istepanian, E. Jovanov, Y.T. Zhang: Guest Editorial Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity. In: IEEE Transactions on Information Technology in Biomedicine. Nr. 4, Jg. 8, 2004, pp. 405–414

Kaletsch, Sunyaev (2011)

Alexander Kaletsch, Ali Sunyaev: Privacy Engineering: Personal Health Records in Cloud Computing Environments. In: ICIS 2011 Proceedings. 2011, pp. 1–11

Lica (2010)

Liviu Lica: Mobile and Social: Ten Best Practices for Designing Mobile Applications. In: Informatica Economica. Nr. 3, Jg. 14, 2010, pp. 60–74

Lin, Ye (2009)

Feida Lin, Weiguo Ye: Operating System Battle in the Ecosystem of Smartphone Industry. In: Information Engineering and Electronic Commerce, International Symposium on. 2009, pp. 617–621

Martínez-Pérez, de la Torre-Díez, Isabel, López-Coronado (2013)

Borja Martínez-Pérez, de la Torre-Díez, Isabel, Miguel López-Coronado: Mobile health applications for the most prevalent conditions by the World Health Organization: review and analysis. In: Journal of medical Internet research. Nr. 6, Jg. 15, 2013, pp. 1–16

Mayer, Weitzel (2012)

Jörg H. Mayer, Timm Weitzel: Appropriate Interface Designs for Mobile End-User Devices–Up Close and Personalized Executive Information Systems as an Example. In: 2013 46th Hawaii International Conference on System Sciences. 2012, pp. 1677–1686

Moore et al. (2012)

Justin Moore, Jeremy Kackley, Paulus Wahjudi, Dia Ali: Android Threat Classification Matrix. In: Journal of Management & Engineering Integration. 2012, pp. 14–19

Nicolaou (2013)

Alex Nicolaou: Best Practices on the Move: Building Web Apps for Mobile Devices. In: Communications of the ACM. Nr. 8, Jg. 56, 2013, pp. 45–51

Njie (2013)

C.M.L. Njie: Technical Analysis of the Data Practices and Privacy Risks of 43 Popular Mobile Health and Fitness Applications. In: Privacy Rights Clearinghouse. 2013, pp. 1–31

Norman (2002)

D.A. Norman: The Design of Everyday Things, 2002

The European Parliament and the Council of the European Union (1995)

The European Parliament and the Council of the European Union: Directive 95/46/EC, 1995. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046: en:NOT, visited on 09/08/2013

Wessels, Purvis, Rahman (2011)

Andrew Wessels, Mike Purvis, Syed Rahman: Usability of Web Interfaces on Mobile Devices. In: Information Technology: New Generations, Third International Conference on. 2011, pp. 1066–1067

West et al. (2012)

Joshua H. West, P. Cougar Hall, Carl L. Hanson, Michael D. Barnes, Christophe Giraud-Carrier, James Barrett: There's an App for That: Content Analysis of Paid Health and Fitness Apps. In: Journal of medical Internet research. Nr. 3, Jg. 14, 2012, pp. 1–11

World Health Organization (2011)

World Health Organization: mHealth: New horizons for health through mobile technologies, 2011. http://whqlibdoc.who.int/publications/2011/9789241564250_eng.pdf, visited on 08/30/2013, pp. I–103

World Wide Web Consortium (2008)

World Wide Web Consortium: Mobile Web Best Practices 1.0: Basic Guidelines, W3C Recommendation, 2008. http://www.w3.org/TR/2008/REC-mobile-bp-20080729/, visited on 09/09/2013

Yeh, Fontenelle (2012)

Shea-Tinn Yeh, Cathalina Fontenelle: Usability study of a mobile website: the Health Sciences Library, University of Colorado Anschutz Medical Campus, experience. In: Journal of the Medical Library Association. Nr. 1, Jg. 100, 2012, pp. 64–68

Erklärung

Hiermit versichere ich an Eides Statt, dass ich die vorliegende Arbeit selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen wurden, sind als solche kenntlich gemacht. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Köln, den 26. September 2013

Curriculum Vitae

Persönliche Angaben

Name:	Phil Diegmann
Anschrift:	Wipperfürther Str. 477, 51515 Kürten
Geburtsdatum:	06.02.1991
Geburtsort:	Wipperfürth
Familienstand:	ledig

Schulische Ausbildung

09/1998 - 07/2002	St. Antonius Grundschule in Wipperfürth
09/2002 - 07/2010	Engelbert-von-Berg Gymnasium in Wipperfürth, Abschluss: Abitur (1,5)

Studium

10/2010 - 09/2013	Universität zu Köln, Wirtschaftsinformatik, B.Sc.

Praktika und Berufserfahrung

02/2007	Krüger Industrieautomation GmbH, Wipperfürth (Praktikum)
04/2008 - 02/2011	Webergy Internet Software AG, Lindlar (Teilzeit)
02/2012 - 02/2014	Forschungsgruppe Informationssysteme und Lernprozesse, Universität zu Köln (Studentische Hilfskraft)
seit 08/2012	Selbststständig (IT-Beratung, Entwicklung und Design)

Sonstige Qualifikationen und Auszeichnungen

Sprachkenntnisse	Deutsch:	Muttersprache
	Englisch:	Fließend
	Französisch:	Gute Kenntnisse
	Spanisch:	Grundkenntnisse
seit 10/2010	Stipendiat der Studienstiftung des Deutschen Volkes	
seit 06/2012	Sitz im Ausschuss "Schule, Generationen und Soziales" der Gemeinde Kürten	