



BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ



FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS
AND MULTIMEDIA

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

IMPLEMENT ANDROID APPLICATION FOR AUDIO PODCASTS

IMPLEMENTUJTE ANDROID APLIKACI PRO STAHOVÁNÍ A POSLECH AUDIO PODCASTŮ

MASTER'S THESIS

DIPLOMOVÁ PRÁCE

AUTHOR

AUTOR PRÁCE

Bc. JÁN ŠVEHLA

SUPERVISOR

VEDOUČÍ PRÁCE

Ing. IGOR SZÓKE, Ph.D.

BRNO 2016

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií

Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Švehla Ján, Bc.**

Obor: Bioinformatika a biocomputing

Téma: **Implementujte Android aplikaci pro stahování a poslech audio podcastů**

Implement Android Application for Download and Listening to Audio Podcasts

Kategorie: Softwarové inženýrství

Pokyny:

1. Nastudujte teorii a praxi k podcastům, najděte a porovnejte podobné aplikace.
2. Navrhněte a implementujte podcastovací aplikaci. V aplikaci se zaměřte na UX (uživatelskou zkušenost) a oboustranou komunikaci se službou Audeliver.com.
3. Aplikaci otestujte na vhodném vzorku beta-testerů. Zveřejněte aplikaci na Google Play.
4. Pokračujte v implementaci a změnách GUI pro co nejlepší UX. Získejte zpětnou vazbu od uživatelů. Implementujte alespoň dvě funkcionality nad rámec běžných aplikací (například podporu zobrazování a vyhledávání v prepisech podcastů).
5. Zhodnoťte výsledky a navrhněte směry dalšího vývoje.
6. Vyrobite A2 plakátek a cca 30 vteřinové video prezentující výsledky vaší práce.

Literatura:

- Dle pokynů vedoucího

Při obhajobě semestrální části projektu je požadováno:

- Body 1, 2 a část bodu 3 ze zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese <http://www.fit.vutbr.cz/info/szz/>

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

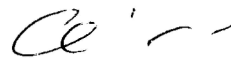
Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Szóke Igor, Ing., Ph.D.,** UPGM FIT VUT

Datum zadání: 1. listopadu 2015

Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 06 Brno, Božetěchova 2



doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstract

This master's thesis is aimed at the problems and issues which are encountered over the development of application for mobile devices. An application for managing and listening of audio podcasts was chosen to be developed. This thesis will guide the reader through general topics and issues of "User Experience" research, its usage and specification on mobile devices and actual design and the implementation of developed application. Results of this work can be used for the development of applications with respect to user centered design.

Abstrakt

Tato diplomová práce je zaměřená na problémy vyskytující se při tvorbě aplikace pro mobilní zařízení. Jako vyvíjená aplikace byla zvolena aplikace pro práci a poslech audio podcastů. Tato práce provede čtenáře základními tématy a problémy "User Experience" výzkumu, jeho využití a specializace na mobilních zařízeních a samotným návrhem a implementací vyvíjené aplikace. Výsledek této práce je možno využít pro vývoj mobilních aplikací s ohledem na "User centered design".

Keywords

Android, application, podcast, UX, user experience, mobile application, design, user research

Klíčová slova

Android, aplikace, podcast, UX, user experience, mobilní aplikace, design, uživatelský výzkum

Reference

ŠVEHLA, Ján. *Implement Android Application for Audio Podcasts*. Brno, 2016. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Szóke Igor.

Implement Android Application for Audio Podcasts

Declaration

I hereby declare that this thesis and the project reported herein is my original authorial work which I did on my own, and that all sources, references and literature used or cited in this thesis were properly acknowledged by complete reference to the source.

.....
Ján Švehla
May 25, 2016

Acknowledgements

I would like to thank my supervisor, Ing. Igor Szóke, Ph.D., for taking on my thesis, inspiring me and overall help.

I also extend my thanks to participants of the user research who helped to develop the best possible experience while using the application.

Last but not least, I would like to thank my family and my friends for supporting me during work on this thesis.

© Ján Švehla, 2016.

This thesis was created as a school work at the Brno University of Technology, Faculty of Information Technology. The thesis is protected by copyright law and its use without author's explicit consent is illegal, except for cases defined by law.

Contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 3 |
| 2 | Theory and Inspiration | 4 |
| 2.1 | Podcasts | 4 |
| 2.2 | Audio-books | 5 |
| 2.3 | Android Platform | 6 |
| 2.3.1 | Google Play store | 7 |
| 2.4 | Existing applications for podcast listening | 7 |
| 2.4.1 | Pocket Casts | 7 |
| 2.4.2 | Podcast Addict | 8 |
| 2.4.3 | Player FM | 9 |
| 2.4.4 | BeyondPod Podcast Manager | 10 |
| 2.4.5 | AntennaPod | 10 |
| 2.4.6 | Summary | 10 |
| 3 | User experience | 12 |
| 3.1 | User-centered research | 12 |
| 3.1.1 | Quantitative research | 13 |
| 3.1.2 | Qualitative research | 13 |
| 3.2 | Design techniques and tools | 14 |
| 3.2.1 | Personas | 14 |
| 3.2.2 | Stories | 15 |
| 3.2.3 | Prototyping | 15 |
| 3.2.4 | Task-flow | 16 |
| 3.2.5 | Usability testing | 16 |
| 3.3 | Rapid Iterative Test and Evaluation method (RITE) | 17 |
| 4 | Specification and design | 19 |
| 4.1 | Problem hypothesis | 19 |
| 4.2 | Audeliver | 20 |
| 4.3 | Interviews | 20 |
| 4.4 | Personas | 21 |
| 4.5 | Stories | 23 |
| 4.6 | Use-case diagram | 23 |
| 4.7 | Main components and wireframe | 24 |
| 4.8 | Flow-diagram | 24 |
| 4.9 | Summary | 26 |

| | | |
|----------|---|-----------|
| 5 | Implementation | 28 |
| 5.1 | Structure of application components | 28 |
| 5.1.1 | In-device database | 30 |
| 5.2 | Advanced features | 31 |
| 5.2.1 | Playback speed control | 32 |
| 5.2.2 | Loading and synchronization of podcasts | 33 |
| 5.2.3 | Audeliver and iTunes search integration | 33 |
| 5.2.4 | QR scanner | 35 |
| 5.2.5 | Audio-books | 35 |
| 5.3 | Used tools and libraries | 36 |
| 5.3.1 | Crashlytics Crash Reporting | 36 |
| 5.3.2 | Publishing | 37 |
| 5.3.3 | Code metrics | 38 |
| 6 | Evaluation and testing | 39 |
| 6.1 | Beta testing | 39 |
| 6.2 | Continuous usability tests | 40 |
| 6.3 | Final tests and users feedback | 43 |
| 6.4 | Future development | 46 |
| 7 | Conclusion | 48 |
| | Bibliography | 49 |
| | Appendices | 50 |
| | List of Appendices | 51 |
| A | Content of DVD | 52 |
| B | Personas | 53 |
| C | Wireframes | 56 |
| D | Usability testing | 59 |
| E | Final feedback | 63 |

Chapter 1

Introduction

Smartphones and mobile devices became an inseparable part of modern lives. People use this new technology every day and maybe every hour. Almost everybody bears with themselves this terminal to the world of uncountable information.

There are many applications easing the access and consumption of a specific content which user wants. One of the most significant factors determining whether an application is successful or not is an intuitive way for users to access the content even without previous instructions or guidelines how to use it.

This thesis aims at the development of a smartphone application for working with podcast feeds. Big focus will be put on user experience research and on cooperation with online service Audeliver. The application has official name AudioCast and will be developed primarily on Android platform.

The first chapter will introduce the reader to the field of mobile devices and podcasts. Smartphone applications became indispensable part of almost every new service for customer segment. The field expanded rapidly and many new technologies and methods for development were introduced. Then terms podcast and audio-books will be explained. They are still not widely known by general public but they have their user base. At the end of the chapter, the existing application for podcast listening will be analyzed.

Chapter two will focus on user experience. This term is relatively new and is firmly building its place also in the field of software development. Some of many definitions, processes, techniques and methods which are used in UX to deliver the best user experience will be introduced there.

UX will remain mentioned also in the fourth chapter. The practical use of UX in design of developed application will be described there. Together with the visual design, the structure and logic of the application will be discussed.

The fifth chapter describes the implementation itself. A reader will be able to read about the main problems encountered during programming, components of application, used technologies and integration with Audeliver service.

Evaluation and user testing will cover the fulfillment of expected attributes of application and goals of UX design. The user feedback regarding the application will be also evaluated. The plan of possible ongoing development shall be discussed at the end.

Chapter 2

Theory and Inspiration

This chapter introduces the reader to the problematics of this thesis. Firstly, the main concepts of the developed application - „podcast“ and „audio-book“ will be described. The targeting platform - Android and its current state will be discussed in section 2.3. At the end, reader can read about similar applications which are already published. Their features and reviews of users will be analyzed.

2.1 Podcasts

Podcasting is a way to publish audio or video recordings on the internet that can be subscribed to using RSS. The term podcast originated from words iPod and broadcasting. It had a steep rise¹ and it is becoming more and more widespread in our society. Despite the fact that the word came from iPod, podcast is not limited to playback on iPod or other devices related to Apple products.

Podcasts are small recordings similar to radio features ranging from few minutes to over two hours. These audio files are stored on the internet and can be downloaded by everybody to listen on demand. Although the very first podcasts were meant to put prerecorded radio programs online, there is an increasing diversity both in the type of files released by podcasters, and in the type of contents [3].

To listen to a podcasts, one can either subscribe to a feed or seek out sites that have links to previously broadcasted programs. Subscribing allows playback devices to regularly download episodes when they are available. In many cases, one can also play previous episodes directly in the browser, either by the application downloaded to his/her device as an audio file or have it played by a playback feature on the website.

Because of simple and widely used technology, download and playback of podcasts can be done on most smartphones or any other devices capable of internet connection and audio playback. It is very likely the device will not support it natively but one can install a separate program for playback (Podcast client or podcatcher) to subscribe to podcasts. Such programs are available for both PC and portable devices.

Jörg Rech [9] states in his works that podcasts are mostly used to publish:

Interviews with experts from industry as well as academia. Topics vary from discoveries in research, distributing information „behind the scene“ or life advices and motivation speeches.

¹Google Trends - <https://www.google.com/trends/explore#q=podcast>

News from the industry informing about new technologies, upcoming events, released products or actual cases in the field. Some are released directly by companies in form of blogs.

Opinions by single expert or a discussion of several person about specific topic. This offers many views and perspectives on complex problematics.

Summaries of conferences including exceptional events, trends, or research results and sometimes even live recordings of presentations.

Educational podcasts which are published by universities or other institutions. It can be recordings from classical lectures or prepared courses i.e. for learning foreign languages.

Music podcasts, mostly with electronic music. Artists broadcasts their live mixing and publish it to their podcast feeds.

2.2 Audio-books

Audio-book (or e-book) is a term for a recording of spoken word. Nowadays it is generally a digital recording but its history dates back to 1931, when they started a program to help vision-impaired people in the USA. Generally, audio-books are based on a written material but it is not always the exactly same content.

Audio-books are distributed in various formats but the most common way nowadays is online. It relies on users if they download it to their computer, smartphone or other portable device. With the rise of portable audio players, audio-books got greater popularity as people listened to it during transportation from one place to another. Common time for listening audio-books is while driving a car. With the rise of smartphones, the audio-books received even greater popularity. Services like Audible² by Amazon provide a big collection of audio-books available to download as audio files or easily via their smartphone application.

The main benefits of audio-books can be summed to following points:

- Visually impaired or illiterate people can learn from books or enjoy fiction literature.
- Audio-books improve listening skills in children and help them to concentrate.
- People who have better sound perception of information than visual perception can learn better.
- Audio-books are generally recorded by professionals who bring voice modality to parts with emotional context.
- Foreign language can be learned or improved in pleasant way.
- Audio-books teach children new vocabulary and help them to construct complex sentence structures.

²Audible - <http://www.audible.com/>

2.3 Android Platform

Android platform was chosen as one of two dominant platforms on the mobile device market for the implementation. Another factor for choosing Android is previous personal experience with development on bachelors thesis as well as following non-academic experience. Development on this platform is possible on any common operating system thanks to multi-platform Java-based Android SDK. Additionally, for testing on real device it was not necessary to obtain additional hardware.

In contrast to Androids biggest competition - iOS by company Apple, Android is distributed under open-source license. At the moment, another competition could be only Windows Phone with increasing market share, however its current coverage does not influence high percentage of usage. Nowadays, Android is used primarily on smartphones and other devices with a touch-screen, such as tablets, multimedia centers, mini computer, smart watches or even cars and some household electronics.

On the other side, the pitfall of Android is the big variety of devices it is operating at. Devices differ mainly by version of operating system, as seen on figure 2.1, but also by resolution of displays. This brings complications with backward compatibility and dealing with limitations of display sizes. These problems are rather well solved on iOS platform because there are only 29 various devices nowadays with 4 main system versions³.

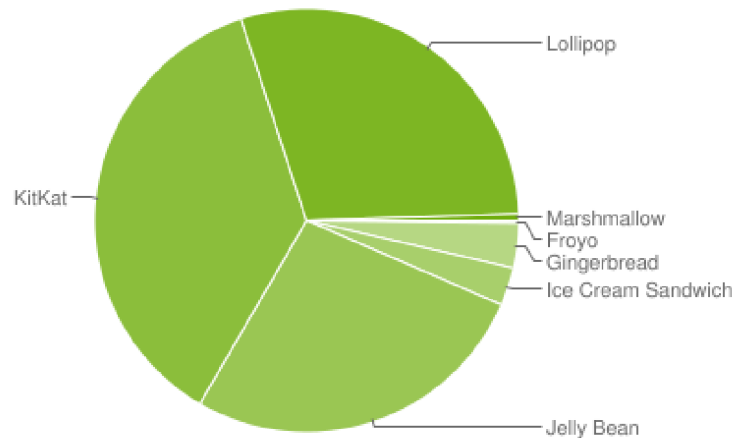


Figure 2.1: Distribution of Android versions. Source: Google Dashboard⁴

Android itself is now developed by company Google Inc., however the whole ecosystem is covered by Open Handset Alliance which is composed of 300 hardware and software companies and mobile operators from the whole world. As mentioned previously, applications on Android are primarily written in Java language. It differs from classic Java primarily by own implementation of virtual machine with name Dalvik. Moreover, Android does not use Java bytecode but Dalvik bytecode. Android Runtime (ART) with version Android Lollipop was introduced there. It should replace Dalvik and bring improvements in performance and responsiveness of applications⁵.

³iOS Version Stats - <https://david-smith.org/iosversionstats/>

⁴Google Dashboard - <http://developer.android.com/about/dashboards/index.html>

⁵ART and Dalvik - <https://source.android.com/devices/tech/dalvik/>

2.3.1 Google Play store

Google Play is an online distribution network which originated on 6.3.2012 by merging Google Music and Android Market. The service allows users to search and download music, magazines, books, films, television programs and applications. Users can also purchase Chromebooks, smartphones, smart watches and other hardware and accessories related to Google. Applications can be downloaded directly to devices that use the Android operating system. In July 2013 had Google Play officially over 1 million applications and over 50 billion downloads [10].

In Android OS, Google Play is represented as a standalone application which allows users to maintain other applications and content. Users can search for specific application or browse categories. If the desired application is a payer, Google Play offers a payment gate. More and more popular ways of monetization of applications is freemium⁶. Users are more likely to download free application and games compared to the priced ones. A basic approach is to offer a free download with limited features or full features for a limited time. Then use an in-app purchase to unlock the full, unlimited application. More advanced approach is to use these micro-payments to offer range of features or content. In games, it can be new levels or playing pieces. In applications, it can be new features which will add new functionality and improve user experience.

In-app purchases can be maintained by developers through Google Play Developer Console. This portal offers a big toolset for developers to maintain their applications. One of the main features is statistics overview which shows how the application is used so developers can optimize the development. Android OS has crash reporting tool to fix problems in released applications, which summaries can be viewed in Developer Console. Other major features are the maintenance of versions, gathering user feedback, monetization, advertisement or access to services and APIs.

2.4 Existing applications for podcast listening

Podcast listening on mobile devices is not a new idea. Therefore there is already a couple of applications with similar functionality and features as developed application in this thesis. This „competition“ makes it challenging to make a high-quality application which will attract new users but also provides inspiration and experience. Google Play provides a platform where users can leave feedbacks about applications. Reviews are public and can be used to learn about aspects of applications which user likes or not. This can be used to design the application better and avoid problems which could occur in developed application.

2.4.1 Pocket Casts

Pocket Casts is the most recommended podcast app by internet media. It is available on all major platforms (Android, iOS, Windows, Web) for \$3.99. Free version is not available. On Android and iOS, the user reviews are overall very good and they praise both wide range of functionality and modern design. Standout features include full playback control from the notification shade, a widget, Google Wear support, variable speed playback, silent

⁶Monetize Freemium Apps - <http://developer.android.com/distribute/monetize/freemium.html>

breaks skipping, filters, automatic cleanup or Chromecast⁷ support. It can be configured to only download podcasts over WiFi, during the night, or when the phone is charging. New podcasts can be added manually, imported from OPML or browsed in big discovery database.

Shortcomings of this application on Android and iOS are minimal and user reviews are very positive. However, the Windows users complain about lack of features. Android users complain about problems with excessive cache size, complicated queuing of local folders, additional feed for web version or some minor bugs.

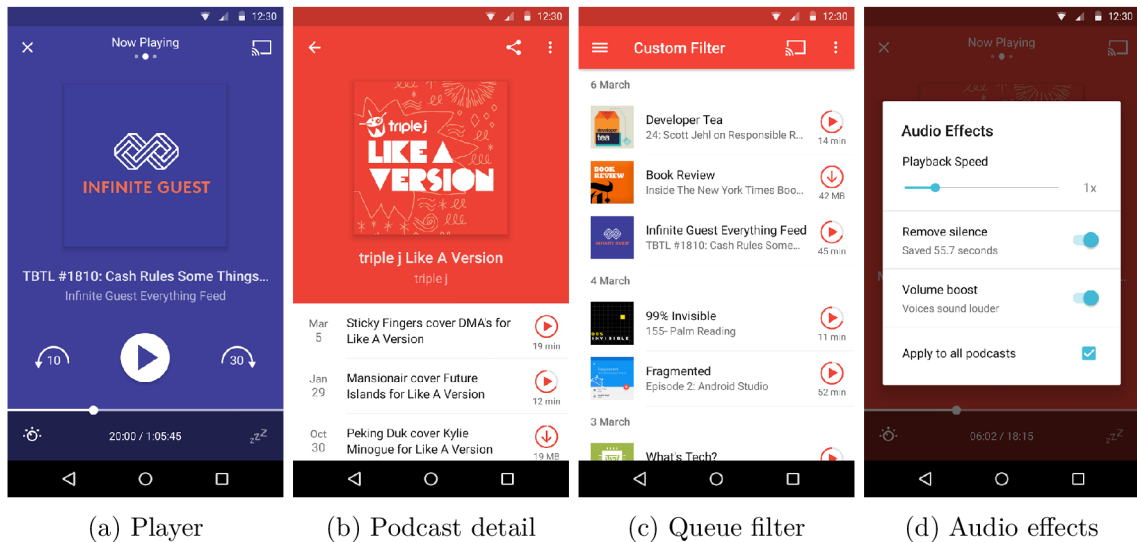


Figure 2.2: Screenshots of Pocket Casts. Source: Google Play

2.4.2 Podcast Addict

Podcast Addict is the most popular podcast player on Android with over million downloads. Developers focus only on the Android platform. It is offered for free with advertisement or without ads for \$2.99. For a freemium application, it is very feature-rich. Its main advantage is great Youtube support. Users can organize and work with Youtube channels as they were regular podcast feeds. Its UI is not designed in the newest material style, but it supports new technologies like Android Wear, Chromecast or integration with Tasker and AutomateIt. Interesting feature is integration with Flattr. Users can donate small amounts of money to their favorite shows using this feature.

The reviews for this application are very good (average 4.6) and author actively replies to them. Missing feature is cross-device sync which is a standard in paid applications. The UI is not very intuitive so an interactive tutorial will be also nice.

⁷Chromecast is a line of digital media players developed by Google. Designed as small dongles, the devices play audio/video content on a high-definition television or home audio system by directly streaming it via Wi-Fi from the Internet or a local network. Users select the media to play using mobile apps and web apps that support the Google Cast technology. Alternatively, content can be mirrored from the Google Chrome web browser running on a personal computer, as well as from the screen of some Android devices.

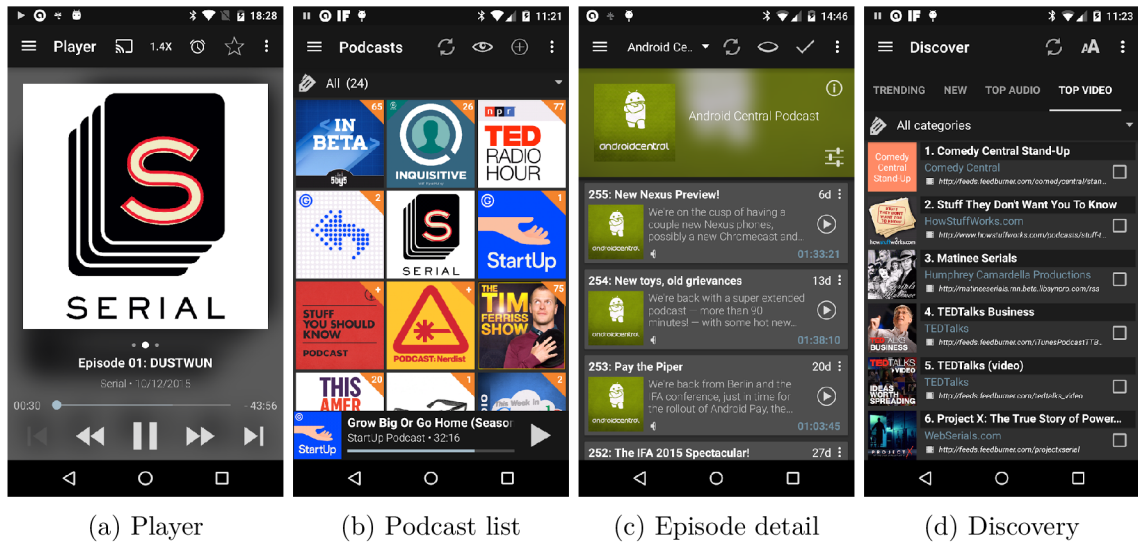


Figure 2.3: Screenshots of Podcast Addict. Source: Google Play

2.4.3 Player FM

Originally a browser-based podcatcher, Player FM has since branched out with its own Android application. Therefore sync with other devices is self-evident. Basic application is for free and some extra features come for a donation. It offers very modern design with pack of user-friendly features. The biggest flaw is its discovery database where new users can easily find podcast based on their interests or previously subscribed feeds. However, this comes with limitation off adding only public feeds. For subscription and download private feed, which will not be added to database, user needs to have paid version. The application has all useful features like import and export OPML, playback speed control, Chromecast and Android Wear support.

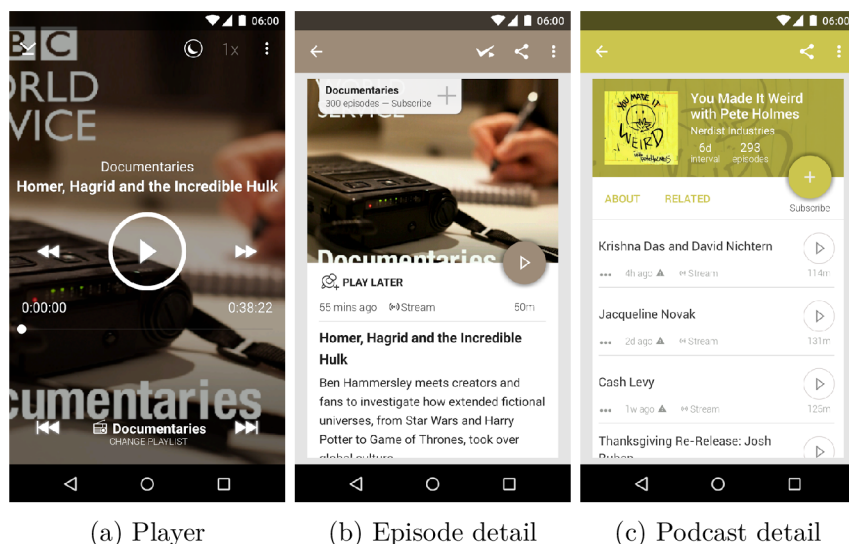


Figure 2.4: Screenshots of Player FM. Source: Google Play

2.4.4 BeyondPod Podcast Manager

One of the oldest podcast players bundles a solid set of features. Its price is among the highest with \$6.99. There is no free version, only a 7-day trial. Its main merit is Smart Playlist generator that automatically lines up shows based on habits of listening. Design is original and does not merge with modern material design. Furthermore it includes homescreen widgets, Feedly integration, ChromeCast support and tools for categorization for users dealing with a lot of podcasts and consuming a broad spectrum of content.

BeyondPod has the lowest average rating with 4.0. Users complain about changes in design, poor UX and removed features. Reportedly, there are also bugs with playback and with downloading of podcasts. The UI is not intuitive, but it contains a short tutorial at first start.



Figure 2.5: Screenshots of BeyondPod. Source: Google Play

2.4.5 AntennaPod

AntennaPod is the only open-source application in this selection. It has minimalistic design in Holo style. Naturally, the application is free to download and also does not contain advertisement. Own database for discovering new podcasts is not available but the application can query search results and categories from iTunes and gpodder.net. The import of custom feeds and OPML support is implemented too. Organizing of podcasts is straightforward, sorted by date when it was added. Users can choose which podcasts to download or a bulk-download. Automatic download is not enabled by default, but offers selection of enabled WiFi networks. Nice feature is support of Flattr for small donations to authors.

Major drawback is limited functionality set compared to bigger or paid alternatives. In reviews, users complain about minor bugs coming with the updates but on the other hand praise the clean UI and simple usage.

2.4.6 Summary

There is more than 20 applications for downloading and listening podcasts in Google Play. They provide various functionalities and features. Five of them were researched thoroughly. Research was focused on non-basic features - which are the most common and which are unique. As seen in table 2.1, applications differ in their features and also in their ratings.

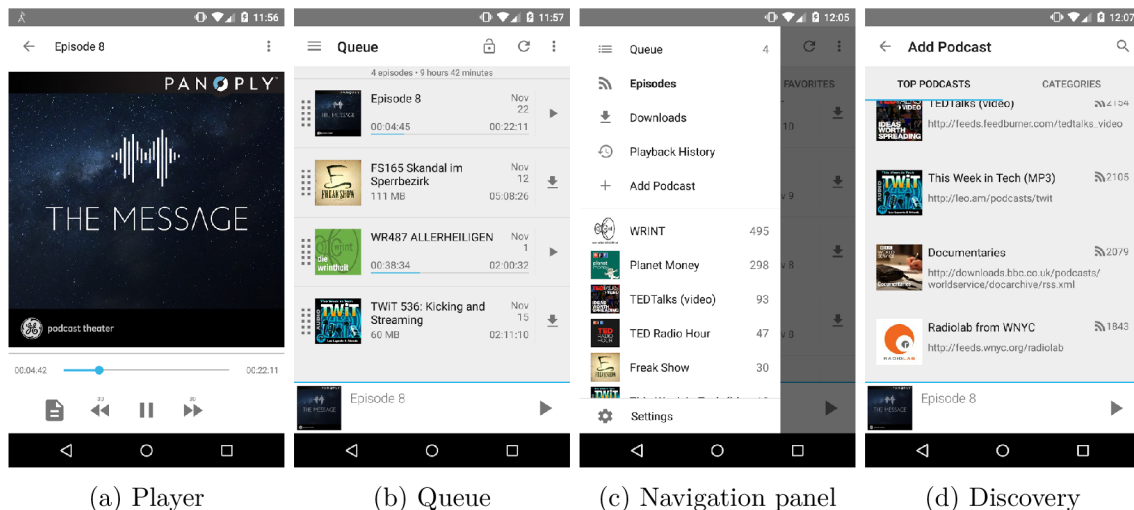


Figure 2.6: Screenshots of Antennapod. Source: Google Play

From the user reviews, it was found that applications share common features like discovery and search in a database of podcasts. With the exception of open-source project AntennaPod the applications support control of playback speed and usage of Chromecast to broadcast the audio. From the table 2.1 can be seen that AntennaPod obviously lacks some features of bigger applications. On the other side, BeyondPod is the most expensive application but its development is stagnating and has very few updates. It is reflected in reviews and ratings of users. Other big applications extended their functionality to Google Wear devices which brought them positive reviews.

| | Pocket Cast | Podcast Addict | Player FM | BeyondPod Podcast Manager | AntennaPod |
|-------------------------------|-------------|----------------|------------|---------------------------|------------|
| Price | 2.99€ | Free/2.55€ | Free | Trial/7.72€ | Free |
| Overall rating, total reviews | 4.6/35.928 | 4.6/174.078 | 4.4/15.661 | 4.0/30.481 | 4.5/4.629 |
| Playback speed | Yes | Yes | Yes | Yes | No |
| Design | Material | Holo | Material | Custom | Custom |
| Local files | No | Yes | Yes | Yes | No |
| Audio-books | No | No | No | Partially | No |
| QR code import | No | No | No | No | No |
| Background synchronization | Yes | Yes | Yes | Pro version | No |

Table 2.1: Features of 5 chosen applications for work with podcasts in Google Play Store

Chapter 3

User experience

Term user experience (UX) is much more complex than terms usability and ergonomics which are often referred as synonyms. Standard ISO 9241-210-2010 [4] defines it as perceptions and reactions of a human being, which are the consequences of using a product, system or a service. User experience includes users' emotions, opinions, preferences, perception, mental and psychological reactions, behavior and achievements while using a product, system or a service. User experience is consequence of presentation, functionality, performance of system, interactive behavior and ability of an interactive system to assist the user. It does not include only how to work with the product but also availability of the product, customer service, support and others.

In the beginning of this chapter the reader can read about main approaches of user-centered research. In section 3.2, UX techniques used in this work will be described. At the end of chapter, the methodology RITE is discussed.

3.1 User-centered research

User research belongs to basic premises for creating a product which by simple and elegant way helps to fulfill needs of users who use it. Leah Buley [1] defines user research as learning as many things as possible about users and their motivation so designer can design a product which can satisfy their need.

Key to a successful product is thus engagement of users to whole process of making and especially to process of creating user interface. To this purpose, there has been formed many methods and practices. It starts at learning from publicly available researches on the internet, goes through questionnaires or asking friends and ends at direct observation of users in environment where they will be using developed product.

Kim Goodwin [2] divides user research according used methods to qualitative research and quantitative research. Typical example of qualitative research can be an interview or direct observation. On the other hand, a typical example of quantitative research is a survey. Each type of research is suitable for different purposes. It will be better to gather data from big amount of surveys or previously done, publicly available research to find demographic data about users. More suitable methods will be the qualitative on the contrary, to better understand behavior of future users.

3.1.1 Quantitative research

Methods of quantitative research are used to analyze the big amount of users. Typical problems which can be solved using this techniques are i.e. the most used operating systems or devices, age categories of users, demographic statistics of users.

The good source of information is, in the beginnings, the publicly available researches of third parties, which can be found on internet, marketing segmentation of market or various analytical tools like Google Analytics¹. In the case of desired data are nowhere to find it is time to compose an own survey.

Survey is the most frequently used and the most representative method of quantitative user research. Kim Goodwin [2] recommends rather to ask short questions and plan the survey distribution well.

3.1.2 Qualitative research

According to Kim Goodwin [2], qualitative research is used to get information about how users behave, how they think about specific activities and which factors influence their behavior.

The most frequently used method is the direct interview. Similarly as when creating survey, it is important to identify the audience and goals. At the beginning, it is good to approach friends or family who can provide contact to representatives of a target group. It is very suitable to use responses from respondents of survey to find users from target group.

Leah Buley [1] calls this kind of approach Guerrilla user research and recommends a few steps to improve the interview:

1. Use organization or personal network to find research participants.
2. Inform participants at the beginning what will the research be about and what will it be used to
3. Ask open-ended questions. Unlike as in a survey, the goal is getting people to open up and share relevant information about themselves without leading them into answer.
4. Ask about past events. Ask to recall the experiences chronologically, which can help them remember specifics that they might otherwise gloss over.
5. Be confident. If the interviewer seem like knows what he is doing, it will put the interviewee at ease.
6. Getting into their environment. The things that people do and keep in their environment are often the biggest source of realization and inspiration for UX practitioners.

The last point can be taken as a separate method. The thorough observation of users can bring most detailed and important insights. It can bring unique information how users behave in their natural environment, what activities do they do, what hacks did they develop to ease these activities or what role does designed product represent in their live. This method is however very time-consuming and it is not certain to obtain all necessary information as Kim Goodwin [2] says. Next complication is understanding and interpretation of what researcher observes. That is why this method is often combined with interview so the participants can explain why they do the actions.

¹Google Analytics - <https://analytics.google.com/>

3.2 Design techniques and tools

User experience consists of many methods used in improving the user experience at all phases of research and design. They help designers to design the product with users in mind, detect the positive experience of interactions with the developed product and enhance it or detect the crucial and risky parts or product which could bring negative experience. This section will describe methods used in this thesis.

3.2.1 Personas

Personas are the primary method used in User Centered Design process which was defined by Donald Norman [8]. Personas are fictitious, specific, concrete representations of target users. Their main use is to support user-oriented thinking. They are created using beforehand collected data. Persona, which is based on thorough analysis, has ideally its own face, name, interests, education, occupation and ideally everything what has a real user. Personas are intended to serve designers as real potential users to remind designers when users are not in front of them. They should not become an excuse not to work with real users.

Tom Brodie

Tom has 8 years of experience in lube shop operations. He's married with two young kids, and his wife jokes that the last time his hands were completely free of grease was on his honeymoon 5 years ago. At the shop he manages, Tom constantly puts out little fires. He works on the floor most of the day, trying to be everywhere at the same time although he prefers to act as greeter and cashier.

Most shop trends get measured on a monthly basis, since Tom has to meet sales targets defined by the owner, Eddie, in order to get his manager's bonus. On a daily basis, Tom frequently monitors car counts, ticket average and employee productivity (especially individual service statistics). Sometimes his team needs a kick in the pants, but he tries to lead by example.

Tom's Goals:

- **Keep the cars coming.** Tom has to rely on Eddie's marketing efforts but car count is his make-or-break figure; he focuses on customer service to generate repeat customers.
- **Reduce labor percentages without sacrificing customer service.** Staffing is a tricky balance between keeping the shop's labor costs down while ensuring employees get enough hours and bay times stay low.
- **Meet or exceed last year's numbers for this month.** The Owner's sales targets aim for year-on-year increases across the board, but in the current business climate Tom is happy simply meeting last year's numbers.

"Sometimes I'm so busy fighting alligators that I forget about draining the swamp."

DE VISE

Persona copyright ISI, inc.

Figure 3.1: Sample of persona. Source: Shlomo Goltz²

Creating personas involves identifying the critical behavior patterns and turning them into a set of useful characterizations. Kim Goodwin [2] in her book addresses 70 pages to personas and recommends these steps:

1. If there are clearly defined roles among respondents, begin the process by comparing the interviewees in only one role at a time.

²A Closer Look At Personas: What They Are And How They Work (Part 1) - <http://www.smashingmagazine.com/2014/08/a-closer-look-at-personas-part-1/>

2. From data, identify behavioral variables - ways in which user behavior differed - and any demographic variables that seemed to affect behavior.
3. Map the interviewees against the variables, then look for people who cluster together across multiple variables.
4. Formulate explanations for that clustering to see if it really is a valid behavior pattern, then keep looking for any other patterns.
5. Once the patterns are exhausted within a given role, do the same for the other roles.
6. Turn each behavior pattern into a persona by articulating goals and adding detail from the data.
7. Fine-tune the personas as a set by clarifying the distinctions among them.
8. Consider whether there is a need for any other personas for political reasons before reviewing rough drafts with stakeholders.
9. Finally, prioritize the personas and develop the narrative and any other artifacts needed to describe them.

Figure 3.1 shows an example of persona of a shop manager. Personas can and should be used throughout the whole creative process and can be utilized by all parts of a software development and design team, and even entire companies.

3.2.2 Stories

User story describes something that the user wants to accomplish by using the product. They originated as a part of the Agile and Scrum development strategies, but for designers they mainly serve as reminders of user goals and a way to organize and prioritize how each screen is designed. User stories should have three main characteristics:

- They are descriptive, but short. User stories do not capture every detail about the users, their background, their ideas and motivations.
- User stories define the user type. They do not dive deeper into the personality but just focus on the character traits relating to usage of developed product.
- They outline one of the main needs or problems that the developed product solves

User stories are often based on personas but do not include so much detail about the user. Example of a user story could be: "As a podcast listener I want to see new episodes of my favourite podcast and download the best ones."

3.2.3 Prototyping

Prototypes can be represented by following types:

Sketches are hand-drawn prototypes which are created during brainstorming by developers, designers, UX experts or other stakeholders. They capture the first ideas in general way and does not include many details. It is recommended to create many alternatives as this is the least time-consuming method.

Wireframes define content - which objects will be contained in the product, functionality - how will the product behave and partially design - layout of objects. Special effort is spent on arrangement of elements, labeling and planned interactions. They do not include pictures and are consisted mostly from lines (wires) and text. Colors are used only to distinguish hypertext links. Wireframes are usually the first output which is presented to the client. It is desired to do all changes and improvements on wireframes.

Storyboards are comic strips that illustrate the series of actions that users need to take while using developed product. They help designers to create empathy with the consumer. Storyboards translate functionalities into real-life situations.

Interactive prototypes are usually partially implemented layouts. They do not need to be implemented using the same technology as the final product. Interactive elements like buttons or form fields should be functional and execute relevant action. It is generally expensive to implement functional prototypes so changes and modifications should be done in wireframes.

3.2.4 Task-flow

Task-flow diagrams illustrate the motion of a user in designed system. User enters in specific spot, does desired actions to accomplish his goal and leaves the system. These diagrams are based on user stories and storyboards. They represent how does the environment change depending on executed actions and how it can navigate the user to his/her goal. It is the users perspective of the product organization. Diagrams should help to identify which spots of the environment will be used the most, which will be crucial and which could be improved or redesigned. The goal is not to connect the whole system but to create the typical paths of users in the system.

3.2.5 Usability testing

User Experience Testing has several names — UX testing, usability testing, user testing. They all refer to the process of understanding what users do and why they do it. It is based on a simple principle. Respondent performs a list of tasks using the product being tested. Observers watch actions, measure time, count number of clicks, count mistakes and take notes. Important part is also interviewing the respondent and taking notes. Data gathered like this provide valid feedback. It helps to identify problems which people have with a specific interface and reveals difficult-to-complete tasks and confusing language.

When researcher assembles the testing tasks, it is useful to use data and outputs from methods of stories 4.5 and flow-diagram 4.8. Kim Goodwin [2] states that different studies showed that usability testing is not so reliable method as many believe. However according to Jacob Nielsen even 5 respondents in usability testing can discover 85% of problems.

Traditional or summative usability testing involves recruiting representative users, defining key tasks, determining a test protocol, testing with users utilizing think-aloud protocols, and recording observations. Brainstorming solutions to problems and re-testing or redesign work feed into the next design cycle rather than into the current usability test phase. Collaboration between stakeholders may also be quite limited.

Steven Krug introduced the approaches [5] approaches which focus on rapid and iterative usability testing in his work. He defined the following tenets:

- If you want to have a great product - testing is inevitable.
- Testing with event single respondent is 100% better than no testing.
- It is much better to do testing with single respondent in the beginning of development than to test at the end with 50 respondents.
- Generally, selection of „right“ respondents is overrated.
- Testing should form an opinion. Not to prove it or refute.
- Testing is needed to be done more than single time - iterative process.
- Nothing can substitute reaction of real audience.

Usability testing by Krug can be conducted and completed in a single day by using only three or four participants and replacing a formal report with a 1-hour debrief meeting shortly after the last participant has concluded their session. As a result, instead of taking a week or two to produce and deliver a report, the practitioner leads a discussion on the observations in the test and asks stakeholders to document the changes that they will make. Then the debrief meeting concludes. The practitioner spends no more than 30 minutes documenting the changes that will be made, sends it out, and it is over. Not only can the usability practitioner move on to the next research study, but the design and/or development team can make changes immediately without having to wait weeks for a formal report as can be the case with a traditional study.

3.3 Rapid Iterative Test and Evaluation method (RITE)

Development of modern software is mainly driven by agile methodologies. However traditional usability testing was designed as a separate process which was executed parallel with development. The standard approaches of usability testing focus on effectiveness and coverage but not so much on needs of commercial development as shipping an improved user interface as rapidly and cheaply as possible. The common problems why discovered issues are not fixed are listed below:

- Discovered usability problems are not believed to be important by the stakeholders.
- Fixing the feature is not easy. It is often more pleasant to develop a new feature than fixing a problem in existing one.
- Results and findings are delivered to stakeholders with delay - after decisions are made. This could be problem if the development moved forward in a feature which UX testing found as wrong.
- It is uncertain if the proposed solution will actually fix the problem and bring expected results. The lack of verification that the fix is working forces the team to implement the usability recommendations on faith, rather than the demonstrated history of accuracy.

Rapid Iterative Test and Evaluation method (RITE) [7] was defined by game developers in *Microsoft* and was used to develop parts of popular game *Age of Empires 2*. Like traditional usability tests, it involves representative users which engage in real tasks with

the product. Test sessions are observed by the product team stakeholders. Users „think aloud“ as they interact with the product. Test sessions are recorded and observations are discussed and agreed upon by all product team stakeholders.

Unlike traditional testing all stakeholders are involved in prioritizing problems observed and identifying solutions to usability issues. Moreover, problems are identified, iterated upon and fixed quickly within the current usability test phase.

Chapter 4

Specification and design

This chapter describes specification and design of developed application. It is based on knowledge from previous chapter. In the beginning it will mention the process how the specification will be developed. In the section 4.3, it will tell about interviews with users which helped to understand the target audience. Findings from interviews are then processed using methods of UX research discussed previously in sections 4.4 and 4.5. Based on these results, the structure and logic of the application is then designed.

4.1 Problem hypothesis

Design is the art of solving problems. One cannot hope to craft a good product if the team does not understand what problem is actually being solved.

Nowadays, customer software is generally developed using agile methodologies which are based on iterative approach.

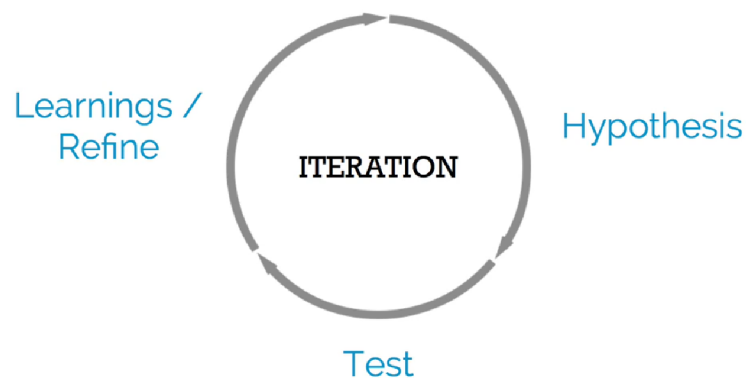


Figure 4.1: Problem solving loop. Source: Maxime Cormier¹

¹Mobile App Design from Scratch: Design Principles, and UX: <https://www.udemy.com/mobile-app-design-from-scratch/>

This paradigm can be followed also in phase of design and it starts with setting up the hypothesis. First important issue is to define the problem itself. Because developed application will be used by bigger audience, focus on customers is crucial. Therefore these hypotheses will be set:

| | |
|---------------------|--|
| Problem hypothesis | Users cannot easily select and download various audio content they find on internet to their mobile devices. |
| Customer hypothesis | Educated, employed people |

The main problem which the developed application will solve will be to ease the process of adding media content to podcast feeds. Current podcast applications and web-pages serve to purpose of consuming podcast feeds from the internet and playing it back to users. They are limited to content in podcast feeds which are released by podcast authors. Users can only choose which podcasts to follow and which episodes to listen or download. None of them has functionality to create or manage the very content of podcast feeds. When users want to listen to lecture available on YouTube or audio on SoundCloud, they need to use custom applications of video servers even though the data are available. To this purpose application will use service Audeliver which will be described later.

Based on the initial knowledge and rankings of favorite podcasts and their topics, target users will be primary employed adult people, who listen to podcasts i.e. while driving a car or traveling.

4.2 Audeliver

Audeliver² is a service developed by ReplayWell, s.r.o.³. The service makes it very simple for users to add various multimedia content, convert it to audio and add to their custom podcast feeds. The process begins with adding media content. Users can either post URL from various multimedia sites (Youtube, Vimeo, Dropbox, Kaltura, SoundCloud) or upload their own media files from their devices. The files are then in the background being uploaded, converted to uniform audio format. When the conversion is finished and audio file is available, the file is added to chosen RSS feed. Various applications or web pages which work with podcast feeds can then work with the audio file - stream or download and play it.

The service offers 3 podcast feeds and 1GB data storage for free. The audio files are stored for 3 days and the bandwidth is unlimited. Paid version enables to have unlimited number of podcast feeds, adds to 10GB of data storage and the files are stored for unlimited time. The service also has API which can be used to list users podcasts, user data and most importantly to add new content to feeds.

4.3 Interviews

Podcasts are not widely known and popular with general public. When I tried to ask random friends about their use of podcasts, only approximately 1 out of 8 knew what is a podcast. That is why it was decided to focus more on qualitative user research than quantitative. I spread the word about this thesis using social media and found active listeners of podcast feeds on their mobile devices - "power users".

²Audeliver - <https://audeliver.com/>

³ReplayWell - <http://www.replaywell.com/>

| | |
|--|--|
| NAME Peter "I love to learn new things all the time" A man in the tram with headphones, taking notes | BEHAVIOR Subscribes to podcast feeds Other content downloads and consumes manually Adding content is complicated, man- aging queue is complicated |
| DEMOGRAPHICS Owns a smartphone, laptop 16 - 50 years old Married Employed, working "with brain" Wants to learn all the time | NEEDS AND GOALS Adding any content to a queue of downloaded files easily Listening to content anytime, offline |

Table 4.1: Proto-persona

Individual meetings were arranged and respondents interviewed about their usage of podcast applications. The questions asked were for example:

- What time during the day do you listen to podcasts and what do you do besides it?
- What is your favorite application for podcasts and why? Did you try alternatives?
- How do you use the application? During playback and otherwise?
- What is the the feature you miss the most in the application?

Interviews with 4 different active users took place and according to it requirements for the system were created. The interviews were also used as a base for 4 personas which can be found in appendix B. Personas will remind the users and interviews during the whole process.

One of interview participants mentioned that she uses podcasts also for music listening and that in Drum and Bass community podcasting is commonly used channel to distribute music from artists to their fans. This fact changes the initial customer hypothesis. Therefore it will be modified it to reflect new facts.

| | |
|---------------------|--|
| Problem hypothesis | Users cannot easily select and download various audio content they find on internet to their mobile devices. |
| Customer hypothesis | Educated working people Fans of music distributed through podcasts |

4.4 Personas

To form the personas, a canvas inspired by LeanMonitor⁴ will be used. This template consists of 4 blocks and should contain these fields:

⁴LeanMonitor - <http://blog.leanmonitor.com/persona-canvas-essential-tool-customer-development/>

Name and sketch should represent a real person to remind designer that he is dealing with humans and not customer segments. It should contain name, face, place and even a characteristic phrase.

Demographics contains information which is the most representative for the user group this persona represents. The information should be of course related to solved problem. Example fields are age, city, job, etc. but it can also be interesting to add other types of information such as annual income, marital status, etc.

Behaviors : are the activities carried out by users which result in the problem or need that developed product solves. It can be filled by stating how users are currently resolving their needs, what kind of solutions are available and which aspects of these solutions they do not like. It should also contain a list of "painful points", competitors and requirements of the product/service.

Needs and goals respond to why the user performs the previous actions. They cannot be generic or abstract. They are the most critical points because designers tend to think they know what are the needs and objectives of user.

This template will be used to form the proto-persona. Proto-persona is a summary of all personas. It originate from brainstorming workshops where designer tries to encapsulate the beliefs (based on their domain expertise and gut feeling) about who is using the product or service and what is motivating them to do so. Proto-personas give to designer a starting point from which to begin evaluating the product and to create some early design hypotheses.

| David | Tomáš | Martin | Lucie |
|---|---|--|---|
| Add YouTube lecture to queue and listen to it later | List the most popular podcasts and subscribe to it | Find a podcast about NLP, select episodes and listen to it later | Choose DnB mix from collection of DnB artists and listen it before sleep |
| <ul style="list-style-type: none"> Views video in YouTube application Transfers it to the application Adds it to Audeliver feed Sets the episode to download Returns to YouTube application Is notified when audio is ready | <ul style="list-style-type: none"> Opens application Finds list of the most popular podcasts Browses it and view detail Subscribes to it Optional: Sets offline queue size | <ul style="list-style-type: none"> Opens application Finds search feature Enters keyword and process Browses podcasts and select episodes to download Finds them later in queue | <ul style="list-style-type: none"> Opens application Opens list of episodes of DnB podcast Finds the desired one Streams episode (without downloading) Sets sleep timer and lets playback turn off |

Table 4.2: User stories

4.5 Stories

Stories were created based on personas and observed behavior with using podcast applications. They are about users and their goals which they want to accomplish using developed product. Stories help to discover the best journey through the application to accomplish the most common actions.

Firstly, stories about personas heading to their goals were stated. Then the single steps how they would achieve the result were described in more detail. These sequences helped to find the crossings of user journeys. This started the thinking process about the structure of the whole application - which screens will be presented and which features will be required. User stories of participants can be seen in table 4.2

4.6 Use-case diagram

Based on the stories and understanding of target user, a use case diagram could be created. Application will have only one type of user - standard user who wants to listen to his podcasts or add episodes to his own podcast.

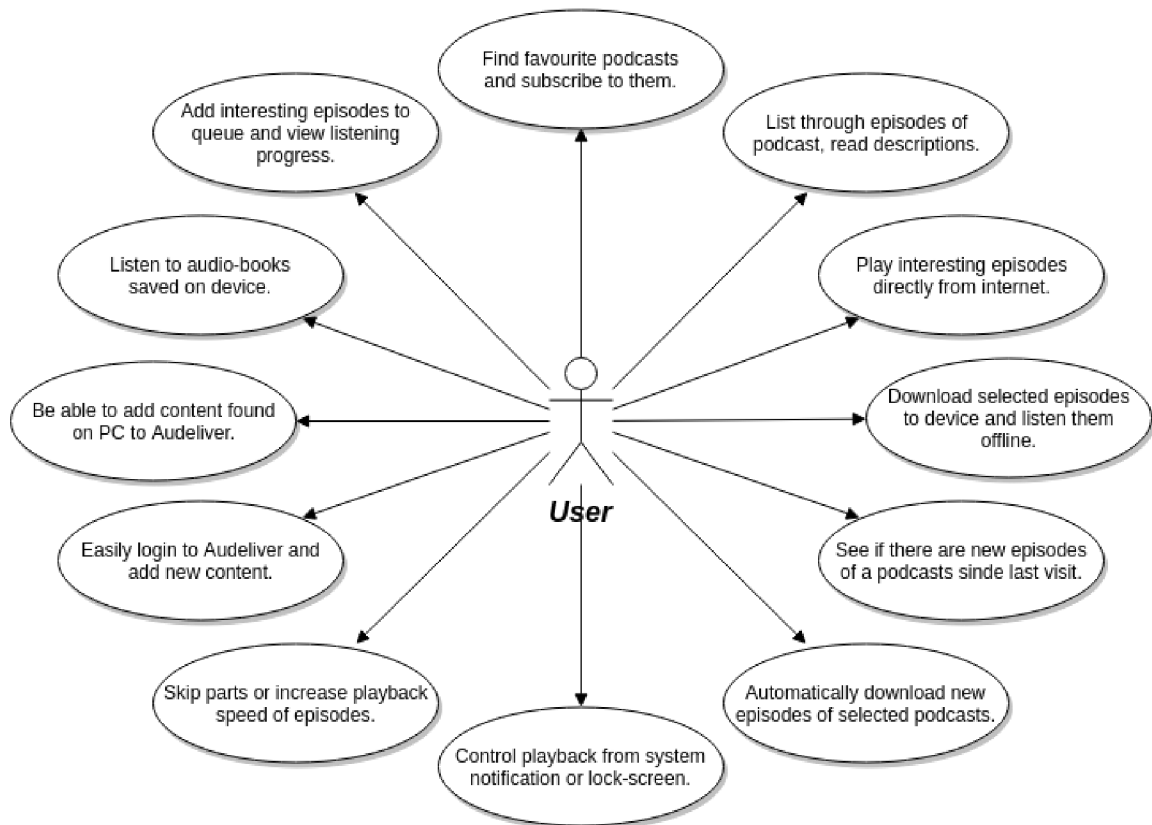


Figure 4.2: Use case diagram of developed application

4.7 Main components and wireframe

After the knowledge about target users was gathered, sketching of possible design of application was commenced. The result of it were the following main components:

Podcasts is a simple list of subscribed podcasts. Here user can open detail of podcast. There will be more options like subscribe/un-subscribe, play episodes, download episodes, etc.

Discovery is a place to search for new podcasts and or import content from outside of application. User can list the result of their search or add content to their Audeliver account.

Play later Is a queue of episodes from different podcasts which will user finds interesting and want to differentiate from the original podcast. Play later will have most of functionality as standard feed - playback of whole queue, automatic downloads etc.

Audio-books Is component where users can upload local audio-books stored on their device. Many audio-books have structure similar to podcast - they consist of multiple audio-files (chapters) so audio-books will be represented as podcasts and chapters as episodes.

When paper sketches got more realistic look, they were transfer to digital form. For this purpose NinjaMock.com⁵ wireframe editor was used. It enables to easily create wireframe model of mobile applications with use of pre-packed UI components and icons. There is also a feature to add on-click actions to regions of wireframe what can be used for interactive prototype user testing.

The final wireframes can be seen in appendix B.

At first, two versions of layouts were modeled - one with a navigation drawer on the side of the screen and one with tabs on the top of the screen. These interactive wireframes were sent to respondents for feedback and everyone confirmed that layout with side navigation drawer is more intuitive and more easy to use.

4.8 Flow-diagram

When the main layout was determined, more detailed structure of the application and the logic of screens was defined. It is the crucial part in UX design and usability. Users want to get expected result from the action or easily revert it in case of a mistake. This topic was frequently asked in usability testing. Thus during the development and many iterations the hierarchy of screens made a few changes. Following figure 4.3 shows the final hierarchy and flow of screens.

As you can see, **Podcast list** is the entry and also the last screen of the application. If the application is launched with a valid intent, it is redirected to corresponding screen but back navigation brings user to the **Podcast list**. Another important screen is **Podcast detail** which can be opened from screens containing list of episodes. In this case the initial screen is added to back-stack and back navigation returns to the last screen.

Application contains navigation drawer which can open all top-level screens. It can be opened from any screen except **QR scanner** and **Player**. When a screen is opened like this,

⁵NinjaMock - <http://ninjamock.com/>

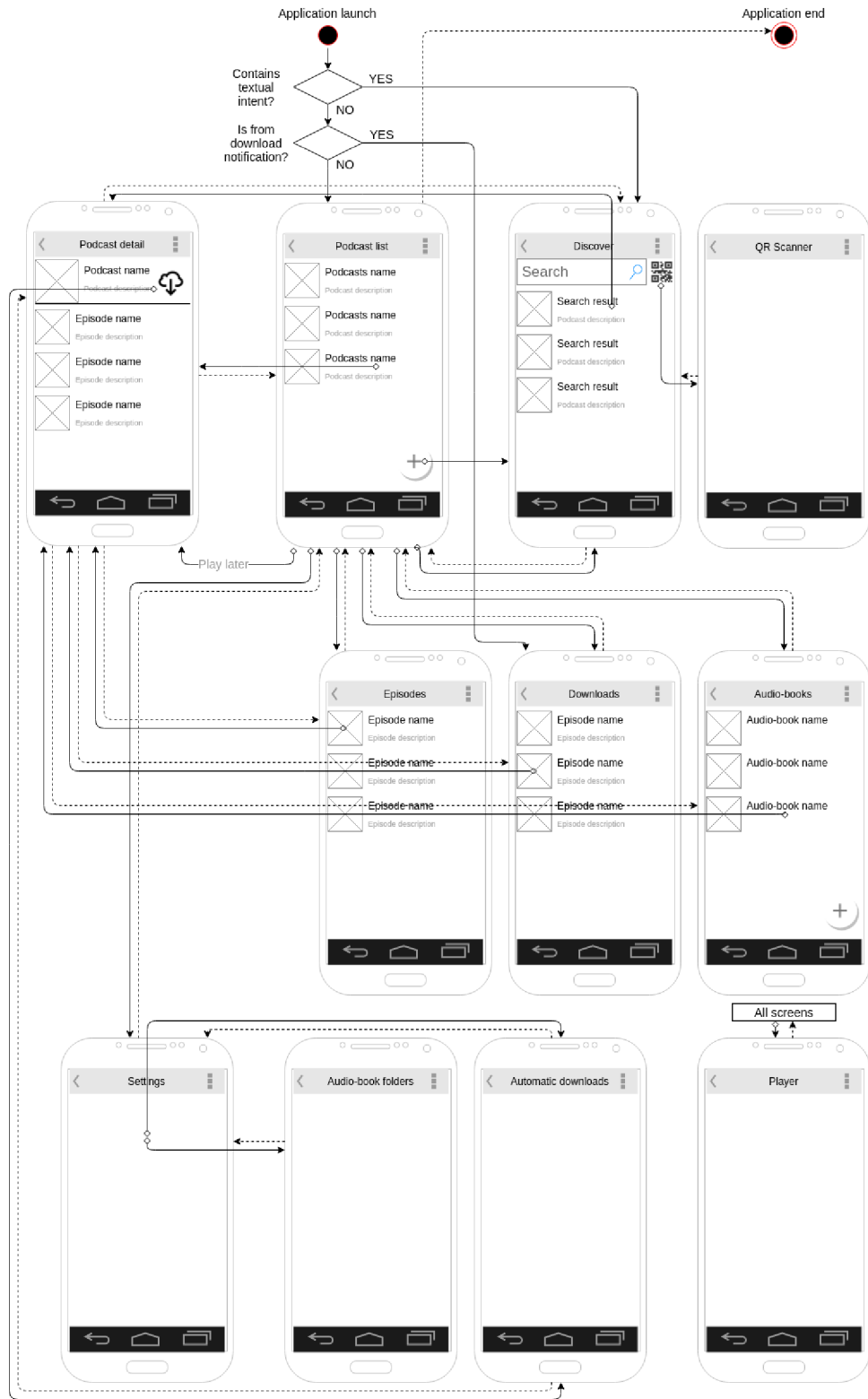


Figure 4.3: Application flow diagram of application containing the most important components

the back-stack is cleared and only *Podcast list* remains. The *Player* screen can be also opened from any screen except *QR scanner*. It is done using mini-player at the bottom of screen. In this case the initial fragment is added to back-stack.

Navigation drawer enables users to login or manage Audeliver account. This action opens a dialog over any screen. When the dialog is closed, the screen below it is returned to front.

AudioCast contains also a tutorial which is shown over specific screens when they are opened for the first time. The tutorial can be also executed individually at any screen. Due to complexity these elements were not included in the diagram.

4.9 Summary

After interviews with users and frequent rethinking of the whole problem these hypotheses came up:

| | |
|---------------------|--|
| Problem hypothesis | Users cannot easily find and play various audio content they find on internet or on their devices. |
| Customer hypothesis | Educated working people Fans of music distributed using podcasts |

These points were used to focus on the true problem.

As mentioned previously, 4 main components were to be used in the application. Decision to use the navigation drawer on the side of the screen enables to use swipe-able list items.

Based on the data from interviews, the most important features were selected and ordered by priority. The list was defined before development and the decisions about the development were made according to it. This list, however, was not changeless. Based on the findings from the iterative usability tests and according to RITE methodology, the feature plan was altered.

Frequently requested features were sleep timer and playback speed control. Both are often implemented in other podcast applications. The sleep timer is straight-forward and easy to implement. Playback speed control is not a standard function in music-playback libraries and required more work. Both these features were added to a feature-list.

Playback of local files (ie. audio-books) was also demanded, however this feature would require a lot of implementation. Moreover it is not in the original scope of podcast application and Audeliver will provide this functionality by uploading files to the feed. It was planned for the later stage of development and released as the last feature.

Podcast discovery and rankings were also mentioned in the interviews. It is a key feature in all applications for podcast listening and was implemented in the early stage of development. It used publicly available iTunes Search API 5.2.3 for searching in database of podcasts by entered keywords.

Another requested feature was grouping of podcasts with similar topics. This feature was planned as component *Boards*. It was intended to work locally (offline) with option to upload custom collections to remote server and make it public for other users. This feature was partially implemented in the early stages of development. It contained demonstrating "server" data and presented to users in iterative usability tests. Due to the negative feedback of power-users and extensive development, this feature was temporarily removed from the application.

During the iterative usability tests new feature requests came up. To work more comfortably with larger lists of episodes users requested an option to sort episodes by different criteria. Another added feature was saving of listening progress. The progress is saved when users skip playback to another episode and when they return to it, playback will continue from the last position. Both these features were implemented in the later stage of development. The option to sort episodes by listening progress was also added.

Chapter 5

Implementation

Based on data about distribution of Android version mentioned in introduction 2.3 and on defined target audience, it was decided to support minimal SDK version 16 which gives more than 90% coverage of all Android devices. The application was developed in Android Studio IDE and Git was used for version controlling. The base structure of application was inspired by templates of developer Petr Nohejl¹. In the application were used the newest available libraries and tools. List of libraries is mentioned inside the application, in *Credits*.

The application is held in Material Design². To that purpose project relied on Android Support library³ and libraries providing Material UI elements.

5.1 Structure of application components

Main functional components of most of Android applications are activities, fragments and services. Traditional Android programming recommends to use one activity for one screen. Activity handles "live" data and resources in current screen, GUI, life-cycle as well as communication with other components. Later it was recommended to move most of logic to fragments which are encapsulated in single activity but can be reused in others. Many modern frameworks and guidelines however use only single activity and only replace fragments with different screens. This is useful for smaller applications because activity will be a single entry point to the application and logic can easily and quickly decide what content to show. Activity also can store shared resources, be used for communication between fragments or handle asynchronous tasks. This approach was also used in AudioCast.

As mentioned, application is based on templates of Petr Nohejl which were modified and extended in my previous projects. As seen on class diagram 5.1, activities and fragments use inheritance chain author reuses in his projects. *TaskFragment* inherits from standard Android fragment and adds methods to execute and process asynchronous tasks with regards to fragments life-cycle. *ContentFragment* adds methods to execute API-calls, process responses and show visual place-holders and indicators of progress over content while tasks are being executed. In AudioCast were added methods to work with *FloatingActionButton*, handle back key pressed, provide title of screen, etc. *ContentServiceFragment* extends it with added methods for binding to desired service. I use bound services with Messenger to

¹Android-Templates-And-Utilities - <https://github.com/petrnohejl/Android-Templates-And-Utilities>

²Material Design - <https://www.google.com/design/spec/material-design/introduction.html>

³Support library - <http://developer.android.com/tools/support-library/index.html>

communicate with fragments or activities⁴. **ContentEpisodesBaseFragment** is AudioCast-specific abstract class which contains list of **EpisodeEntitys**, adapter for **RecyclerView** and shared methods to work with list of episodes and sort it. Thanks to general scope of **EpisodeEntity**, all fragments listing episodes can inherit from it. **EpisodesFragment** and **DownloadsFragment** serve for sole purpose while **PodcastDetailFragment** is used for feeds of podcasts, **PlayLater** queue and detail of audio-book as well.

Dialog fragments are used for interaction with Audeliver so they can be shown over any screen. Dialog fragments were chosen over standard dialogs so the API calls are done in the dialog self. Like this it is easier to handle asynchronous calls and show appropriate placeholders.

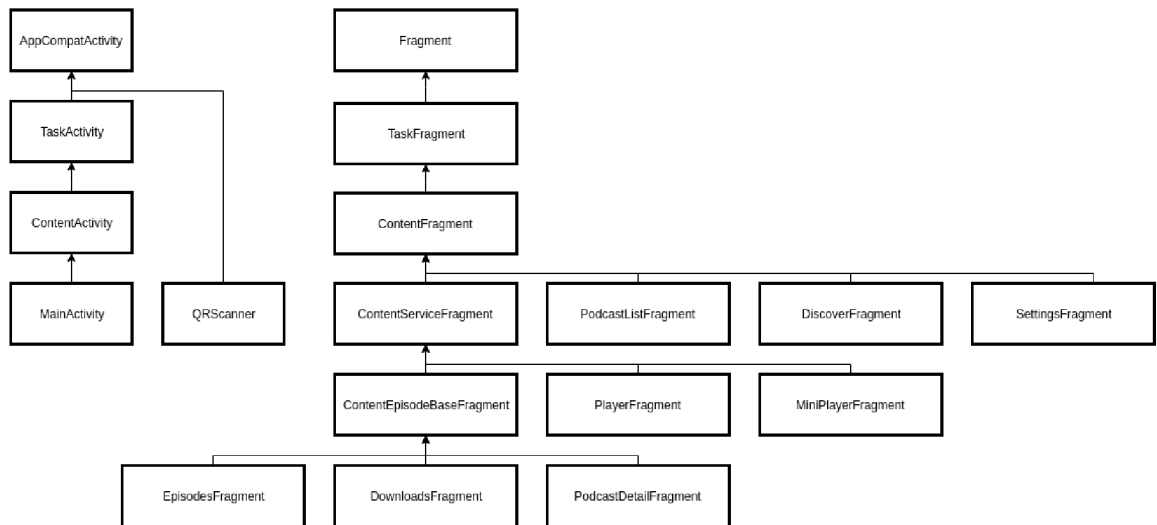


Figure 5.1: Class diagram of most important classes in application

Services

Identical hierarchy of inheritance is used also in activities, **Services**. AudioCast uses two services:

DownloadService handles downloading of episodes in background, Audeliver uploads and synchronization. The progress of downloads is propagated to currently bound fragment using messages. Fragment then updates its adapter with episodes. For Audeliver uploads there is no push-notification support so pull strategy was used. After the Audeliver upload is added, the service continuously calls API and current state of the upload. Synchronization executes asynchronous tasks to update database. It is triggered from custom **BroadcastReceiver**. User can choose under which circumstances the synchronization will happen. If the conditions are not suitable a new **BroadcastReceiver** is started. In **AndroidManifest.xml** it is declared to register internet connectivity changes and power changes. When user conditions are met, it triggers the synchronization in service again.

PlaybackService obviously handles playback of audio files. For media playback Android offers simple interface **MediaPlayer**. It was used it in the beginning of development

⁴Bound services - <http://developer.android.com/guide/components/bound-services.html#Messenger>

and it worked great. It can easily play local files and also stream files from remote server. However its interface is limited and does not have a lot of options. Because of users' demand for playback speed control **MediaPlayer** was replaced with **ExoPlayer**⁵. It is an open-source project officially maintained by Google. Its audio and video components rely on **MediaCodec** interface which was released in Android 4.1 (API level 16). This does fit API limits of AudioCast.

PlaybackService is not a binded service like **DownloadService**. To control it from fragments there is a static class **PlaybackRemote**. It holds reference to the service using **MediaControllerCompat.TransportControls**. During initialization it injects to the service object of **PlaybackBase** which extends **MediaSessionCompat.Callback**. Thanks to this system classes the **PlaybackBase** is controlled also from **MediaNotification** and thus also system-supported media controls like A2DP bluetooth devices. **MediaNotification** is shown when the playback starts and is set as foreground so it cannot be closed. When playback stops, notification remains shown but can be canceled by user.

Activities

As mentioned before, AudioCast uses only two activities. **MainActivity** is the entry point to the application. When the application is opened, **MainActivity** initializes services, navigation elements and shows initial fragment. If the application was opened using **Intent** from other application and it contains relevant data - text with URL - **DiscoveryFragment** is shown with URL as a initial value. If the application was opened from notification, corresponding fragment is opened. **MainActivity** is also responsible for showing **FloatingActionButton** and distributing its events to **ContentFragments**, distributing back-press events to fragments, changing fragments and holding fragment-stack.

QRScannerActivity can be opened only from **DiscoveryFragment** and serves to single purpose which will be described later.

5.1.1 In-device database

Android offers multiple ways of storing persistent data. AudioCast uses 3 of them:

SharedPreferences API is an interface to access and store simple key-value text pairs using XML files private disk space of application. In AudioCast it is used to store simple fields as user settings, progress of tutorial,

SQLite database is the default Android database. It has some limitations compared to traditional SQL implementations but it is more than sufficient for storing AudioCasts entities described below.

Device storage is used to save downloaded audio files. Default location is applications default files folder **sdcard/Android/data/com.audiocast/files** but can be changed by user.

Typically using SQLite in Android required a lot of boilerplate code, which takes considerable time. There are many open-source libraries to ease the use of databases and reduce redundant code. **Sugar ORM** was decided to be used. It encapsulates class of desired entity and automatically creates database table. It uses data types and field names

⁵ExoPlayer - <http://developer.android.com/guide/topics/media/exoplayer.html>

according to Java class definition. The class must inherit from provided **SugarRecord** class which implicitly adds **id** variable and thus also column. Inherited instance methods serve to save, update and delete object. Static methods provide interface to operations with whole table like selecting records, updating, counting, deleting and executing custom SQL queries. These methods provided all necessary operations used with AudioCast data model. Changing and updating of database scheme is inspired by migrations in Ruby on Rails. **AndroidManifest.xml** file contains version of database beginning from 1. When number of version is increased, Sugar ORM searches in **assets** folder for SQL script file with the same name as the number of version. It is executed only once.

Data models in AudioCast are rather simple and are designed for re-using and unified access across application. Two main models which are also saved in database are **EpisodeEntity** and **FeedEntity**. **EpisodeEntity** represents every audio track in application. Standard use is for episodes of podcasts but the model serves also for files of local audio-books. This decision was made because of unified approach to audio files. Like this, audio-book files can be easily added to play-later queue and played the same way as podcast episodes. **MediaPlayers** interface plays files from URL which can be either local file or a link to remote file.

FeedEntity represents either Podcast feed (including Audeliver feeds), as well as local folder with audio-book. **UserEntity** is self-explanatory. Fields are parsed from Audeliver login/register response and **AuthToken** is generated from combination of email and password. Android database is itself well-secured, so no additional security arrangements were necessary.

ER diagram of AudioCast database is shown on figure 5.2

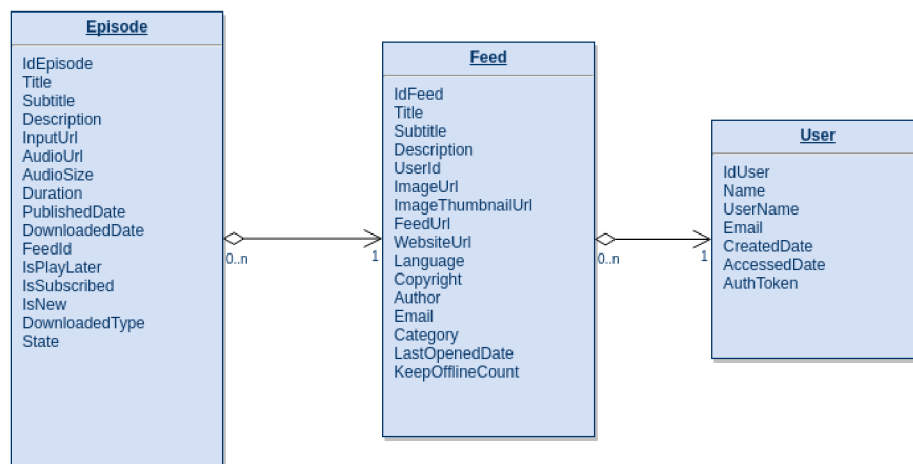


Figure 5.2: Entity relationship diagram of AudioCast database

5.2 Advanced features

Part of the assignment was to implement a podcast application and at least two advanced features. I implemented more of them and are described below.

5.2.1 Playback speed control

One of the most frequently requested feature by users was Playback speed control. Standard **MediaPlayer** interface does provide easy interface and the most important features. However, playback speed was not one of them. With Jelly Bean (4.1 - API 16), Android was provided with new APIs for low-level media manipulation, like **MediaExtractor**, **MediaCodec** and **MediaCrypto**. Using these, one can define behavior of each part of player like networking, buffering, extraction, decoding or rendering. However to implement the whole player it requires a lot of work and code.

ExoPlayer is project that implements these APIs. Its sample demo application has a nice wrapper that provides a higher level interface but still enables to re-implement parts of player. Figure 5.3 shows the high level object model for an ExoPlayer configured to play MP4 streams. It was used it as a base for custom implementation.

Core of **DemoPlayer** is object of class **ExoPlayer**. This class handles internal state of player but does not implement the playback self. After initialization, this functionality is injected in a renderer object. AudioCast does not need video renderer as in MP4 so only custom **VariableSpeedMediaCodecAudioTrackRenderer** was injected. For **SampleSource** it uses default **ExtractorSampleSource** and **DefaultUriDataSource**. The playback speed control self is implemented in renderer.

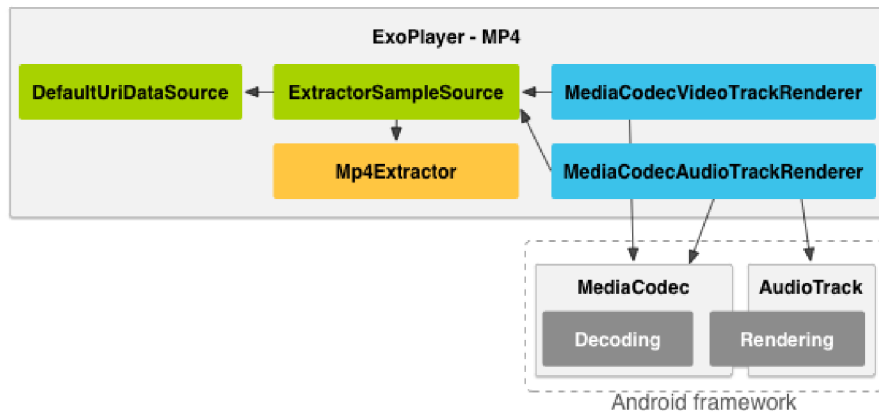


Figure 5.3: Object model for MP4 playbacks using ExoPlayer

This feature is not standardly implemented in music players and only a few podcast/audio-book players successfully implement it. Therefore the resources are rather limited. One of the recommended options for Android is to use **SoundPool**⁶. However, it is designed mainly for short audio sounds and with longer audio tracks, the performance and stability showed to be very poor. After a comprehensive search open-source library for speeding up or slowing down speech **Sonic**⁷ was discovered. It is primarily implemented in ANSI C but also in pure Java. The Java implementation uses only standard components so it can be easily used in Android. The CPU intensive work is done with integer math and performance is really good even on older devices. ANSI C code was also tried in AudioCast, using **sonic-ndk**⁸ and Android NDK and JNI. However the quality difference was not noticeable. For speeds below 2X it uses PICOLA algorithm with time-domain harmonic scaling.

⁶SoundPool - <http://developer.android.com/reference/android/media/SoundPool.html>

⁷Sonic - <https://github.com/waywardgeek/sonic>

⁸sonic-ndk - <https://github.com/waywardgeek/sonic-ndk>

5.2.2 Loading and synchronization of podcasts

Podcast feeds are available in XML files stored on remote servers. These files can be extensive and contain a lot of data. Plus, depending on the size of audio files and internet connection of server it can take a long time to parse and process the whole feed. In the beginning of development AudioCast enabled to process only 20 latest entries from feed so the time required to parse all subscribed podcast was tolerable. However, this limitation was not acceptable by users so parsing of whole podcast was implemented. Majority of podcasts publish in their XML feed circa 100 latest episodes. While using application I encountered podcast with circa 300 episodes. When user is subscribed to circa 10 podcast, the processing takes around 2 minutes. This is not an action user wants to wait for in foreground. Therefore it was moved to **DownloadService**. It should be done automatically and in periodical intervals. User is able to set frequency of the synchronization.

One way was to initiate the synchronization is to check time periodically in service using a **Timer** however it was CPU consuming and not reliable. A solution of using **BroadcastReceiver** with **AlarmManager** API came out. **AlarmManager** can be used to run a **PendingIntent** on specific time. Moreover, the **BroadcastReceiver** can be defined with intent filter **android.intent.action.BOOT_COMPLETED** which will run it with device boot-up.

Synchronization consists of two phases - synchronization of feeds and synchronization of downloads. To parse the feeds AudioCast uses **Earl**⁹. It is a lightweight library which parses both RSS and Atom feeds and produces simple POJO. The values are then processed and copied to **FeedEntity**. AudioCast then parses episodes and either merges them with existing episodes in database or creates new ones.

AudioCast can download latest episodes of podcasts. User specifies how many should be downloaded. When new episode is published, the older ones are deleted. To download episodes AudioCast uses **FileDownloader**¹⁰ library. Initially used Androids native **DownloadManager** was used but similarly as with **MediaPlayer**, it provides limited interface. It did not enable access to current progress of downloads or pausing of download which are both important features.

5.2.3 Audeliver and iTunes search integration

Audeliver provides a simple XML API. To communicate with it AudioCast uses mechanisms in asynchronous tasks. To parse it to entities is used **SimpleXML**¹¹ library. It matches the structure of XML to entity using annotations on variables of class. It can be used to parse all podcasts however to unify behavior of feeds AudioCast processes Audeliver feeds from RSS XML. Custom API calls are used to authentication requests - login and register, for acquiring list of Audeliver feeds and for uploading of new recording.

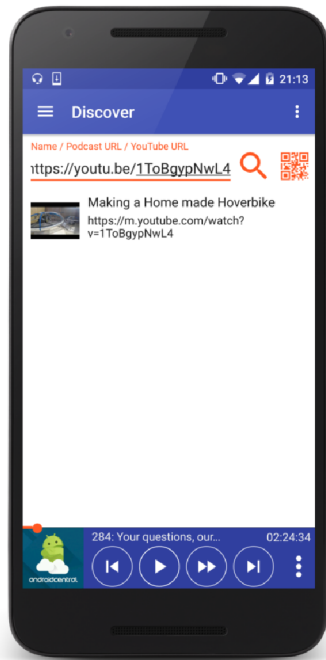
Audeliver does not support push notifications when upload is finished, so in **DownloadService** AudioCast periodically calls API for progress of upload. When it is finished, corresponding feed is refreshed and task is done.

To simplify adding of content to Audeliver a „sharing of video to AudioCast“ feature was implemented to AudioCast. It uses **intent-filter** defined in **AndroidManifest.xml**. This intent-filter states to Android system that the application can receive specified type of **Intent** from other applications. Intent-filter in AudioCast is set for textual data. It is

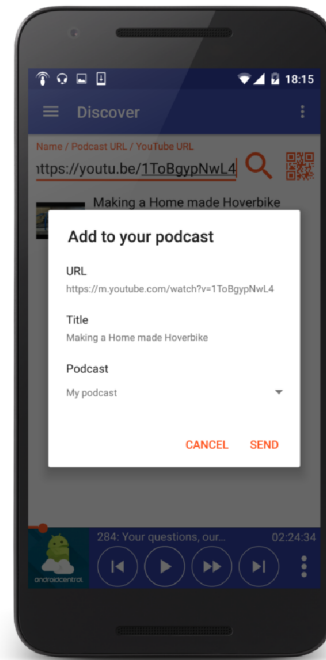
⁹Earl - <https://github.com/einmalfel/Earl>

¹⁰FileDownloader - <https://github.com/lingochamp/FileDownloader>

¹¹SimpleXML - <http://simple.sourceforge.net/>



(a) Parsing a YouTube URL



(b) Dialog to add video to Audeliver

attached to the **MainActivity**. When the activity is started and intent contains relevant data, **DiscoverFragment** is opened with the text as parameter. It asynchronously validates the data. These 3 states can occur:

Valid YouTube URL is then queried by YouTube **oEmbed API**¹². The URL of a video is concatenated with provided oEmbed URL. The response contains basic information about the video, including original title. The image of video is received using **YouTube Data API**¹³. The ID of the video is pasted into given API URL, which responds with default video thumbnail. Like this, the video entry is distinctive for the user and can be processed to Audeliver upload.

Valid feed URL is verified if it is an XML document and if it contains RSS of Atom feed. It is then processed to general podcast loading process mentioned before. If it contains relevant feed and entries, it is displayed as regular podcast and user can view details and subscribe to it.

Other text is considered as a search query and is passed to **iTunes Search API**¹⁴. For this API AudioCast uses the same APICall mechanism as with Audeliver API. It returns a **json** object which contains list of found relevant podcasts. Entries contain fields about podcasts in iTunes database. Feed URL, image URL and title are parsed

¹²oEmbed - <http://oembed.com/>

¹³YouTube Data API - <https://developers.google.com/youtube/v3/>

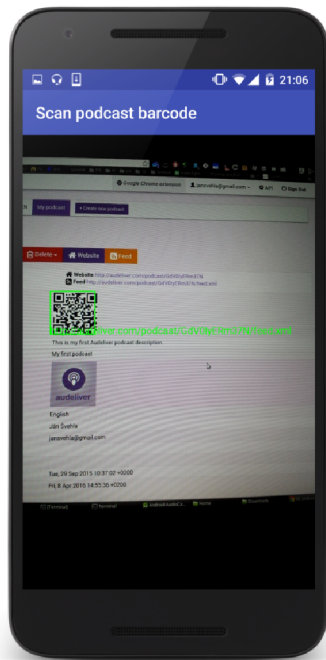
¹⁴iTunes Search API - <https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/>

and used to show list for users. When an entry is opened, AudioCast parses the feed and shows the details - same as with valid feed URLs.

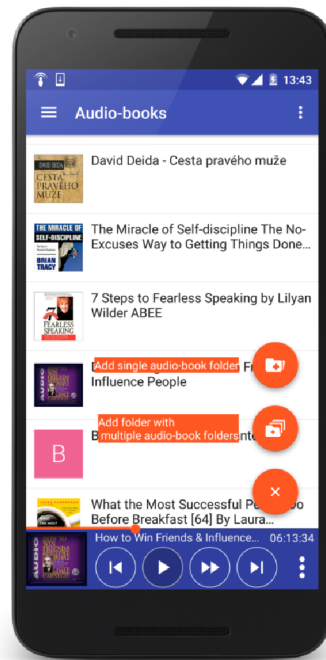
5.2.4 QR scanner

Another feature to improve usability of application was ease the way how to transfer data from desktop computers or laptops to the application. Simple and pleasant option showed to be generating and scanning of QR codes. To implement the functionality new **Barcode API**¹⁵ was chosen. The sample application provides a nice example, which was integrated into AudioCast. Android 6 (API level 23) introduced new **Runtime Permission**¹⁶, which needs to be handled when using device camera. User is prompted to grant permission to AudioCast. If user does not enable it, the feature is disabled.

QRScannerActivity is opened from **DiscoveryFragment** and if it successfully scans a barcode it returns textual result. It is then pasted to **DiscoveryFragment** search field and processed the way described above 5.2.3.



(a) QR code scanner in AudioCast



(b) Audio-book list

5.2.5 Audio-books

In prefatory user research multiple users stated that they would like to use single application both for podcast listening and for audio-books. Both are mostly spoken words and used for education. However, all the competing application for podcast listening which were

¹⁵Mobile Vision - <https://developers.google.com/vision/barcodes-overview>

¹⁶Runtime Permission - <http://developer.android.com/training/permissions/requesting.html>

researched [2.3.1](#) are aimed for listening and downloading of podcasts from internet. Only **BeyondPod 2.5** supports adding local audio-files to custom feed. None of them support is directly.

Thanks to AudioCasts reusable data model, integrating audio-books to application was fairly easy. AudioCast assumes that each audio-book is saved in its own folder and contains multiple audio-files (chapters of book) which have corresponding names alphabetically ordered. User can either choose to import a single folder like this or a folder which contains multiple folders with audio-books (collection). For selecting a directory **NoNonsense-FilePicker**¹⁷ library was used. It is used also for changing of default downloads folder in settings. This library internally handles Android 6 permissions - reading an external storage also requires user-granter permission. When user selects folder, AudioCast recursively scans for audio-files. Using Androids **MediaMetadataRetriever** API it tries to retrieve meta-data as published date or author of audio-files. Each valid audio-file is added to database as an **EpisodeEntity** and each folder as **FeedEntity**. Like this, it is easy to reuse UI elements from podcasts.

5.3 Used tools and libraries

Android development grew significantly in past years. Thanks to is and Googles open-source attitude, there is a big community of developers sharing their knowledge and tools. On a web-based Git repository hosting service **GitHub**¹⁸, there are many open-source repositories for Android project. The amount of libraries is large. To find the right ones for AudioCast **Android Arsenal**¹⁹ portal was used.

For development was used the newest version of **Android Studio**²⁰. During the development there was conducted an update to version 2. One of the new features was **Vector drawables**²¹. Before that all the graphics were rasters and required to be packed with the application in multiple resolutions - for different device screen densities. It meant excess effort for developers. Vector drawables enable to use graphics in vector representation. This feature is enabled only from Android 5.0 (API level 21) but for backward compatibility Android Studio will automatically generate raster images in corresponding resolution for older Android versions. Vector drawables can be created directly in XML or imported from SVG files. Moreover, Android Studio provides extensible collection of default icons. Other icons were downloaded from free icon database **Flaticon**²².

5.3.1 Crashlytics Crash Reporting

Google Play Developer Console provides a way to gather crash reports from installations in production. When the application crashed, a dialog is shown to user with option to report crash to developer. However, user needs to explicitly confirm the action and generally not many users are willing to do it. There are are many automatic crash-reporting tools. They are attached to application and when application crashes, they gather relevant available information and send report to external server. Many developers use open source or ad-hoc

¹⁷NoNonsense-FilePicker - <https://github.com/spacecowboy/NoNonsense-FilePicker>

¹⁸GitHub - <https://github.com/>

¹⁹Android Arsenal - <https://android-arsenal.com/>

²⁰Android Studio - <http://developer.android.com/tools/studio/index.html>

²¹Vector drawables - <http://developer.android.com/tools/help/vector-asset-studio.html>

²²Flaticon - <http://www.flaticon.com/>

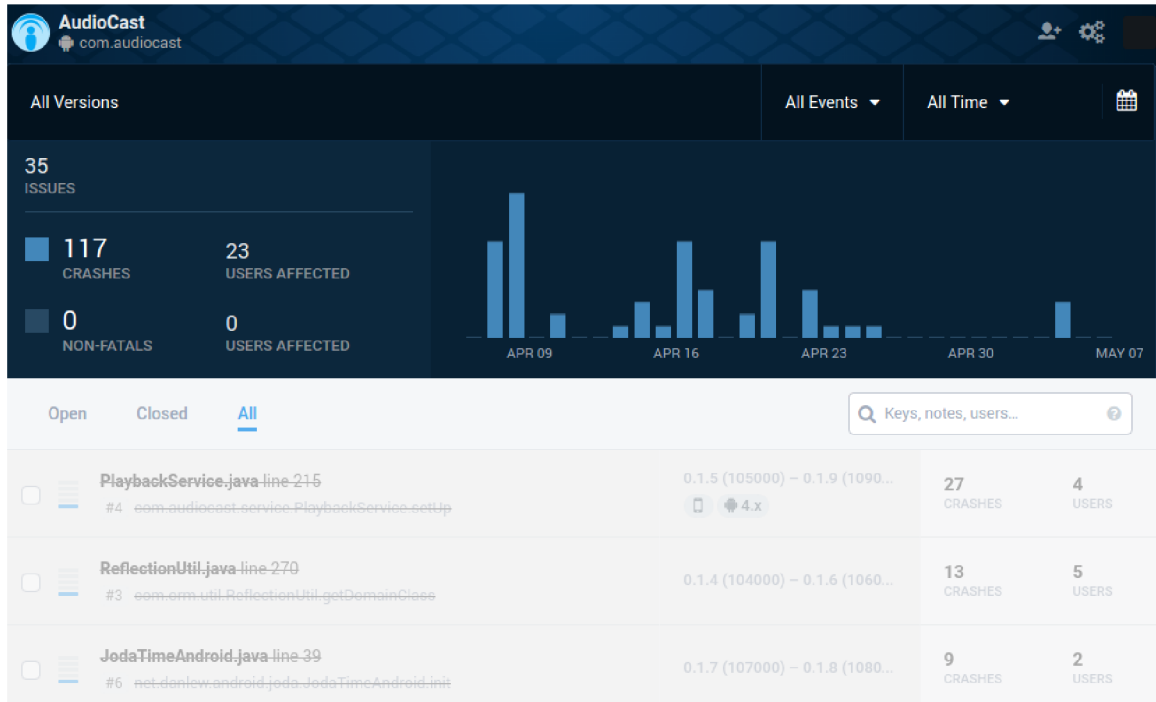


Figure 5.6: View over Crashlytics crash reports

solutions of their own, like those that send crash reports via email, but often quickly become overwhelmed by the amount of data there is to parse.

I decided to use **Crashlytics**²³. It is a royally free service with own servers and web tools to present and analyze results 5.6. It is easily added to application - using a standard dependency and module plugin. Moreover, there is an Android Studio plugin, which automates the whole integration.

It enabled to collect and fix crashes from all devices that were not available during development.

5.3.2 Publishing

AudioCast was published on Google Play Store²⁴. To publish application, one must create a developer account with initial fee of 25\$. Developers can publish only digitally signed applications. The asynchronous key is generated with first publication of the application and can not be recovered in the case of loss. Without it, it is not possible to update the application in Play Store. After successful upload of new version, it is put under review and lasts a couple of hours.

For promoting new versions and building up a community of users, a Facebook page²⁵ was created. Through it, the users were informed about new features, their questions were answered and feedback was collected.

The application is primarily developed in English but it was simultaneously translated to Czech and Slovak languages. Android has a useful system of text management from the

²³Crashlytics - <https://try.crashlytics.com/>

²⁴AudioCast - Google Play Store - <https://play.google.com/store/apps/details?id=com.audiocast>

²⁵AudioCast - Facebook - <https://www.facebook.com/AudioCastApp>

very beginning. All string should be saved in .xml resource files in a key-value pairs. The key is the same for every language - it is an identifier. Values are the strings translated to corresponding languages. All string from each language are saved in saved in single file **string.xml** and stored in language-specific folder in **res** folder of project. If any key is not found in the language-specific file, the default value is used. Default build configuration does not allow to build signed application with un-translated strings.

5.3.3 Code metrics

Application was developed with use of a lot of libraries. It enabled to implement extensive functionality in very short time and with limited resources. Although the code often uses simple API provided by libraries, the metrics of the application are significant. Android Studio has plugin **MetricsReloaded**²⁶ which was used to count metrics in AudioCast. Table 5.1 shows the numbers of lines of code in the application. Module **app** is a custom code developed for this application except the code for QR scanner 5.5a. It was re-used from sample application and represents circa 3500 lines of code. Most comprehensive packages are **fragment**, containing all the logic of the fragments, and **utility**, containing all utilities for working with database, podcast feeds, text processing, etc.

| | Java LOC | XML LOC | LOC | Product LOC | Non-commenting LOC |
|--------------------|----------|---------|-------|-------------|--------------------|
| app | 25345 | 5045 | 31596 | 31316 | 28178 |
| CircleTimerLibrary | 489 | 18 | 534 | 500 | 423 |
| ShowcaseLibrary | 2145 | 281 | 2503 | 2407 | 1646 |
| Total | 27979 | 5344 | 34921 | 31510 | 29910 |

Table 5.1: Code metrics of AudioCast implementation

For assuring the code quality Android Lint²⁷ was used. It helped to discover potential bugs, performance problems and security threats. It is by default used when building signed .apk file for release. These checks prevented to publish untranslated strings or potential problems with XML layouts.

²⁶MetricsReloaded - <https://github.com/BasLeijdekkers/MetricsReloaded>

²⁷Android Lint - <http://tools.android.com/tips/lint>

Chapter 6

Evaluation and testing

The only way to find out if it really works is to test it.

Steve Krug [5]

This chapter will guide the reader through the user testing and continuous improvement of application. In first section 6.1, the reader can read how beta-testing was conducted during the development. It brought many new knowledge but the most useful insights were from continuous usability testing which is described later. At the end of development, an extensive testing was conducted which is discussed in section 6.3. At the end of chapter, there are proposed next steps which could be done in future development.

6.1 Beta testing

Beta testing is a methodology when a new release of application is firstly published to closed group of users. They use the application and report problems they encountered to the development team. This version of application is usually modified to gather more usage data and send it to developers automatically. When creating a group of beta-testers, it is good to create a community where they can discuss and share their opinions. Especially if the testing is on a voluntarily base, testers usually do not pro-actively report problems or give proposals for improvement. The developer team needs to actively ask them to do so.

There are multiple ways how to deliver testing application to group of beta-testers. Natural way is to generate an installation file (.apk file) with release build of application and deliver it to users via email or via shared storage. Testers download the application to their devices and manually install it. On Android-powered devices, the Android system recognizes¹ the .apk in email attachment and shows **Install Now** button which redirects directly to installation. Due to security precautions, Android implicitly does not allow users to install applications other way than officially - through Google Play Store. Installation from .apk file will start only if the user enables **Unknown sources** in device settings.

Google also provides a more pleasant way of distributing application to testers - through the Google Play Store. Developers upload digitally signed application to Play Store separately from production version. There are 3 ways to set up beta-testing:

¹Alternative Distribution Options - <http://developer.android.com/distribute/tools/open-distribution.html>

Open beta testing - developer gets a special URL to Play Store which is distributed to testers. When it is opened, Play Store offers opt-in button to enter beta-testing to currently signed-in Google account. Because the URL is public, this method can be used to spread the testing group rapidly.

Closed beta testing is restricted to closed group of testers. Every testing account must be added by developer in form of email address or .csv file with multiple addresses.

Closed beta with Google+ community is the oldest option. A Google+ community is created for the group of testers. To join this group tester must firstly become a member of the community and then can joint the beta program.

The group of testers then automatically receives updates to the newest beta-version of the application through Google Play Store. The original production version will be overridden this way. Developers must regard proper versioning of application builds because Google Play will install the highest version available (both from beta and production). When the beta-testing is stopped, the group of testers remains but the testing versions of application will be overridden by newest available version.

There are also other commercial tools for distributing and managing beta-testing like **Beta by Crashlytics**². It is over-connected with other tools of Crashlytics like crash-reporting 5.3.

In the early development of AudioCast, the first method was used - early versions of application was sent to testers by email or installed the application directly to their devices. When the application grew, a bigger group of testers would have been useful. For this purpose, beta-testing in Google Play Store was started. Requests were published through authors personal social network. They addressed directly power-users with whom the testing of design in the beginning was made. Together the group of AudioCast testers counted circa 10 people. From the beginning Crashlytics was implemented so all application crashes were reported and could be fixed.

Thanks to this, important problems were discovered before the new version was released to production. The application had problems with specific older devices which had too low heap memory size for loading bigger number of images. This problem was not possible to reproduce with devices which were available. The crash reports from Crashlytics are anonymous and can not be assigned to specific user. Only available info is device model so I contacted group of testers and found the corresponding one. With developer tools and device in hand it was able to fix the problems.

6.2 Continuous usability tests

As mentioned before 3.3, the application was developed using RITE framework with combination of usability tests according to Steve Krug [5]. A big amount of respondents was not available during development so the usability tests were often made with a group of beta-testers. Many of them were power-users so they provided me very useful insights and feedback.

The usability testing sessions were conducted in light-spirit and easy atmosphere. In the beginning I invited respondents to quiet cafes or restaurants. After the introduction

²Beta by Crashlytics - <http://try.crashlytics.com/beta/>

and short small-talk we proceeded to tests. I explained what the session is about, how long it will take and what it is good for. Useful thing to say was that we are testing the application and not the tester so there is no wrong answer. I asked them to think-out-loud - to talk about everything what they see, how do they understand what they see, what they are doing and what they think it will do. To initiate this mood, I firstly asked them about their favorite applications and what they like about them. During this, I also took notes so the respondents do not think I write down only negatives.

I prepared circa 6 different scenarios, which included the most important actions in the application. The scenarios evolved during the development as new features were added. A few specific questions regarding the task were prepared for each scenario and were asked right after the task was finished.

The most insightful reports from testing can be found translated in attachment [C](#).

With the release of the first version usability tests commenced. The GUI of the first version was almost identical with the design on final mock-ups which were created in cooperation with a part of the respondents.

The first version of application could parse podcast feeds and subscribe to them. Users could download episodes of their podcast manually. It could both play locally downloaded audio files and stream from the internet. Application parsed only last 20 episodes of a podcast and the data were persisted as a serialized objects in SharedPreferences [5.1.1](#). Because of small a amount of episodes this did not present a performance problem. It showed list of subscribed podcasts, detail of podcasts with list of episodes and a screen for player.

I conducted a usability testing with one of the power-users. The aim of this test was size of elements One thing is to test UI using a mock - on a PC screen but experience on a real hand-held device is other. Respondent was positive about the layout of the application and overall navigation - using the navigation drawer. There were minor confusions because of low-quality items. Respondent complained about the lack of functionality - which was really minimal in that stage. Main problem was missing search functionality - user needed to find a podcast (using web browser), copy a URL to clipboard and paste it to AudioCast.

This version was also distributed to a group of beta-testers using email. Thanks to their usage, bugs in playback and in downloading of episodes were discovered.

After the initial verification the development continued for a few weeks. The next release contained following improvements and features:

- Screen listing only downloaded episodes.
- Screen listing all episodes of all subscribed podcasts.
- Boards screens - entity for grouping similar podcasts.
- Play-later queue.
- iTunes database search [5.2.3](#).
- Automatic periodical synchronization of podcasts.
- System notification with playback control.
- Integration of Audeliver API.

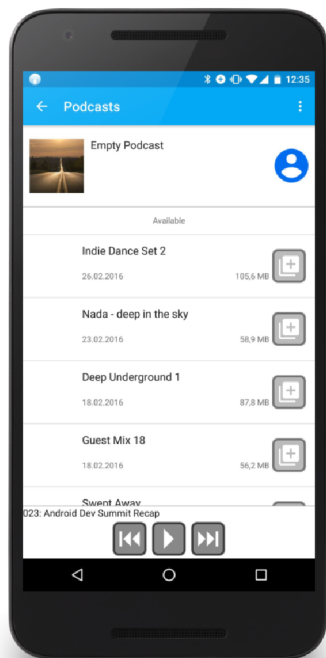
New single features were sent to beta-testers but due to occasional bugs and frequent changes the motivation of testers suffered. Frequency of updates was lowered.

Specific features were tested right after the code reached a stable state.

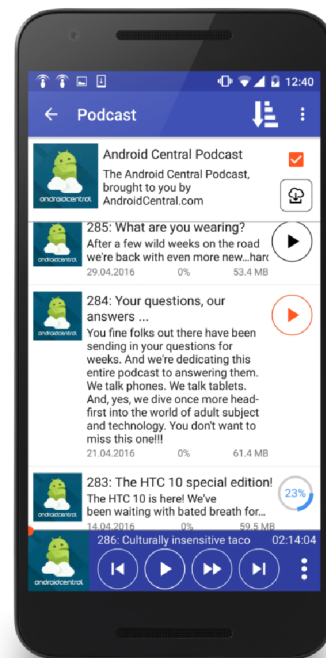
Important changes were made to **Discover** screen. The initial set-up used the text-field just for iTunes search. To add a podcast from URL user needs to click a button and paste URL into a text-field in a dialog. Tested respondents firstly tried to paste a URL to the search box for iTunes search. As a result, the text-field was improved to parse URL strings and used them to directly open detail of parsed podcast. Hint text was added to the text-field and later it was improved to handle YouTube links as well.

After testing **Boards** concept, it was temporarily removed from the application. Intender purpose of boards was as a collection of similar podcasts. User could subscribe to the whole board or just find similar podcasts. These collections should be implemented and stored on a remote server. For demonstration, an in-device hard-coded collections were created. However, meeting with power-users showed that this was not a feature they would rely on or create new collections to share with community. Due to extensive implementation it was decided to temporarily remove this feature.

Audeliver API integration was implemented smoothly. In the beginning the API did not provide an option to register a new account. In the dialog for interaction with Audeliver in AudioCast, there was a button to register a new account however it only opened web-site where the users made a registration. Entering registered credentials again - to login in application was also needed. After this, new API calls in Audeliver were requested and after a short time they were delivered. Registering of a new account was integrated to AudioCast and provided a smooth experience.



(a) Detail of podcast in the beginning



(b) Final version of detail of podcast

The screens containing episodes also came through a change 6.1b. Firstly each episode item in the list contained little detail and main action was to add episode to Play Later. Clicking on the item started playback. The items showed two lines of description and intention was to open detail of episode to see more description. However, respondents in usability testing showed antipathy against this way and wanted to see description more quickly. Entries were then made expandable - clicking on the item expands title and description and shows more detail. The main action was changed to play/pause button which is colored in accent color if the episode is downloaded. The button is replaced with progress indicator if the episode is being downloaded. The podcast detail header of an Audeliver podcast contained a button to open authorization dialog firstly however it was removed because the same action is accessible from navigation drawer and users are familiar with it after they login.

Screen for automatic downloads also came through changes thanks to usability testing. At first, the first item in list of podcasts was dedicated to control overall synchronization. It contained switch to enable/disable periodical synchronization and button to start manual synchronization. This item was visually similar with entries for normal podcasts to fit the style. However, this was discovered as faulty. Respondents had problems with the understanding of the controls. Another problem was discovered with manual downloading of latest episodes. To do this, the application showed two dialogs - one to ask if user wants to "Force synchronization" and second one to choose number of latest episodes. The first one was evaluated as confusing and not intuitive. As a result, the overall switch was moved to **AppBar** - next to a label of the screen. The problematic dialog was removed and the button takes user directly to selecting number of episodes.

Tutorial was tested with users too. Users who were not familiar with podcast applications before welcomed the tutorial with great feedback. Power-users however said that they do not need it and it is bothering them. Resulting action was to re-implement the library and add button to disable all tutorial tips in the application.

6.3 Final tests and users feedback

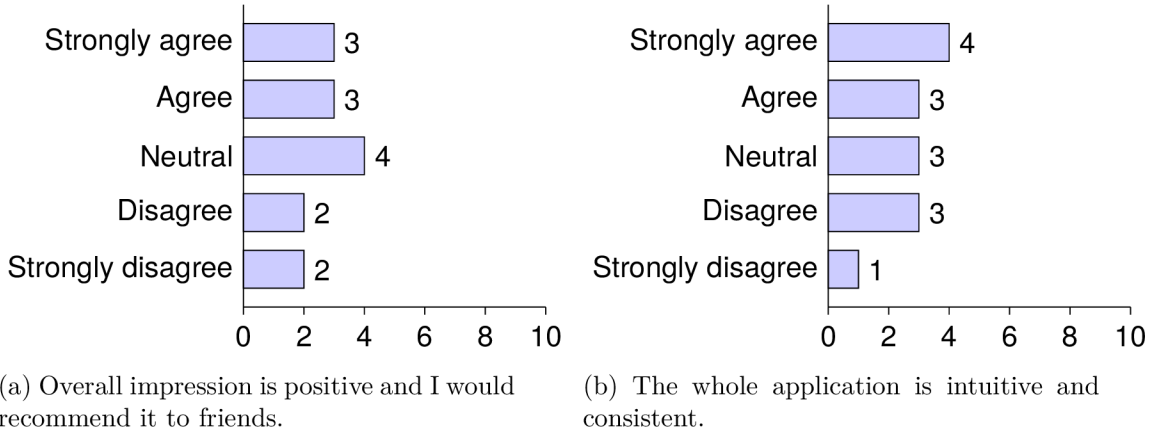
When the application contained most of planned features and development was coming to an end, a comprehensive usability testing took place. A set of actions was prepared which went through the most important parts of application. To each action there were specific questions to get specific knowledge about important elements. Prepared actions and question were following:

- Find your favorite podcast, subscribe to it, find some interesting episodes and play them.
 - Is Discover function easy to find and explanatory?
 - Would you use more details about the episodes?
 - Is the Play Later feature explained in tutorial well? Did you understand what is it good for?
- Set your favorite podcast to automatically download 4 newest episodes. Then download two latest episodes from the podcast.
 - Did you find the automatic downloads easily? Was it intuitive?

- Do you know about more ways how to download latest episodes?
- Create an Audeliver account and find your podcast.
 - Is the login button intuitive?
 - Is register/login process smooth and easy?
 - Where would you expect your Audeliver podcasts to be?
- Change the playback speed of the currently played episode.
 - Is this control accessible? Where would you rather put it?
- Find a YouTube video on your mobile and add it to your Audeliver podcast.
 - Did you know about this feature? Where from?
 - Would you welcome more info about the episode?
- Find a YouTube video on your computer and add it to your Audeliver podcast.
 - Did you know about this feature? Where from?
- Import your collection of audio-books and listen to one.
 - How did you understand the two different add-modes?

At the end of the session, the respondents got a questionnaire about their experience using the application. It was made according to Likert scale [6]. It included the following items:

- The controls of playback in application and in notification are intuitive.
- The player screen is explanatory and has all important controls.
- The podcast detail contains all important information.
- The tutorial is useful and I read it.
- I understand what is a podcast.
- Settings screen is clear and explanatory.
- Automatic-downloads screen is intuitive.
- Register/login to Audeliver is easy and smooth.
- My Audeliver podcasts are in the easy to access.
- Adding content to Audeliver account is easy and smooth.
- Discovery screen is intuitive and easy to use.
- The whole application is intuitive and consistent.
- Overall impression is positive and I would recommend it to friends.



Except for respondents of usability tests, this questionnaire was sent to group of beta-testers and selected people who used the application. Together it was filled by 14 users. Overall results were fairly positive what I find as a great result. Positive feedback was mainly received from power-users and from users who used similar applications before. Less-positive feedback was from users who did not use podcasts before and use mainly basic functions and applications of their smartphones. From these results, it can be concluded that the application should not be aimed at general public. The target audience showed to be correctly estimated during creating of personas 4.1 in the phase of design.

The results of summarizing questions can be seen on chart 6.2b. Other results can be found in appendix E.

The application was published circa 2 months before this thesis was finished. In the meantime it was promoted using authors Facebook profile. I published it to my personal page and to few student groups. Boosted with this minimal marketing, AudioCast had 65 installs from Google Play Store. More than half of the installs remained installed and is being automatically updated to the latest available version. Application has stable audience of daily users. Figure 6.3 shows statistics Crashlytics analytics tool, taken on 23.05.2016. The values are taken only from release versions of the application, debugging usages are not included. The statistics show that AudioCast has stable user base despite the minimal marketing. With the proper marketing it could surely reach much bigger audience.

| | AudioCast | All applications in Music & Audio |
|-------------|-----------|-----------------------------------|
| Android 4.4 | 30.30% | 28.19% |
| Android 5.1 | 25.22% | 13.54% |
| Android 6.0 | 24.26% | 8.68% |
| Android 5.0 | 11.11% | 14.09% |
| Android 4.3 | 6.09% | 4.20% |
| Android 4.1 | 3.03% | 11.30% |

Table 6.1: Google Play Developer Console statistics about Android versions of AudioCast installs

Statistics from Google Play Developer Console 6.1 say that application is installed mainly on Android 4.4 (KitKat) version. Compared to the average distribution of Android version in Music & Audio applications, AudioCast has bigger presence in newer versions of

Android. Android Lollipop (5.1) and Marshmallow (6.0) have bigger usage at the expenses of Android Jelly Bean (4.1). It can be explained by target audience of educated people and by geographic location of central Europe.



Figure 6.3: View over Crashlytics analysis screen of AudioCast

6.4 Future development

AudioCast was developed to a fully functional and tested application with a solid set of features. The development was intense and the most important and demanded features were implemented. During the usability tests, users also talked about various other features they came up with or saw in other smartphone applications they use for podcast listening. They might surely be useful and would improve the functionality of AudioCast to compete with the best application available on market. The features that were mentioned the most frequently and were evaluated to fit the scope of AudioCast are as follows:

Boards

As mentioned in design 4.9, the concept of boards was partially implemented in AudioCast but temporarily removed. This component was intended to work as a tool to share sets of podcasts or find similar podcasts from same category. Selected boards would be uploaded to remote server and each board should have been marked with tags. Tags would be used as keywords when searching through database of boards. To initially fill or substitute the database, gPodder API³ could be used. It provides a big database of podcasts marked with tags and references to similar podcasts.

Full-text search in episode descriptions

Podcast feeds often counts hundreds of episodes. Each of them has denouncing title and generally have relevant description which summarizes the content of episode. These data could be used to search in the collection of subscribed podcasts and easily find episodes containing specific keywords or a topic. To make search results more relevant, episodes could be ordered by number of matches of keyword in the description.

Deeper Audeliver integration

AudioCast integrated Audelivers most important features like registering a new user, logging in, listing podcasts and adding new episodes to podcasts. During the development of application there were added new API calls which were not yet implemented in AudioCast. Option to create or delete a Audeliver podcast and delete an episode would be surely useful. Another feature requested by users was backup and restore of podcast subscriptions to Audeliver account. Few users expected that when they login with their Audeliver account on another device, it will restore their subscriptions from previous device. This feature, however, does not exactly fit the scope of Audeliver service so it might be implemented separately.

Another useful improvement would be to upload audio files directly from smartphone to Audeliver podcasts. Every smartphone possess a solid microphone which could be used to record various audio or even new episodes of a podcast. With simple editing it could be an easy way to publish own podcasts.

³gPodder - <http://gpodder.org/>

Chapter 7

Conclusion

In this thesis, I successfully designed, implemented, tested and published an Android application for podcast listening. The work was strongly driven by user experience research. I used it not only during the design phase but also during the whole development, according to RITE methodology. Each new feature was tested and evaluated by respondents in continuous usability tests.

Thanks to the users experience research and usability testing, the application is intuitive and easy to use what was confirmed during the final evaluation and measurements. For the development I used the newest technologies and many available libraries which enabled to implement a big set of features with limited time and resources.

Integration with Audeliver service gives AudioCast a great advantage compared to other applications which enable users just to consume content from podcasts. I want to continue developing this application and to extend its functionality even more. One of the considered ways would be to implement recording and uploading of audio directly from device which could make it a great option for beginning podcast publishers.

The result of this thesis is a stable application which can compete with the most popular commercial application available on market.

Bibliography

- [1] Leah Buley. *The User Experience Team of One: A Research and Design Survival Guide*. Rosenfeld Media, 1st edition, 7 2013.
- [2] K. Goodwin and A. Cooper. *Designing for the Digital Age: How to Create Human-Centered Products and Services*. Wiley, 2011.
- [3] Dinan Gunawardena, Thomas Karagiannis, Alexandre Proutiere, and Milan Vojnovic. Characterizing podcast services: Publishing, usage, and dissemination. In *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '09, pages 209–222, New York, NY, USA, 2009. ACM.
- [4] Iso. ISO 9241-210:2010 - Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems, 2010.
- [5] Steve Krug. *Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability Problems*. New Riders Publishing, Thousand Oaks, CA, USA, 1st edition, 2009.
- [6] R. Likert. *A Technique for the Measurement of Attitudes*. Number nos. 136-165 in *A Technique for the Measurement of Attitudes*. publisher not identified, 1932.
- [7] Michael C Medlock, Dennis Wixon, Mark Terrano, R Romero, and Bill Fulton. Using the rite method to improve products: A definition and a case study.
- [8] Don Norman, Jim Miller, and Austin Henderson. What you see, some of what's in the future, and how we go about doing it: Hi at apple computer. In *Conference Companion on Human Factors in Computing Systems*, CHI '95, pages 155–, New York, NY, USA, 1995. ACM.
- [9] Jörg Rech. Podcasts about software engineering. *SIGSOFT Softw. Eng. Notes*, 32(2):1–2, March 2007.
- [10] Nicolas Viennot, Edward Garcia, and Jason Nieh. A measurement study of google play. *SIGMETRICS Perform. Eval. Rev.*, 42(1):221–233, June 2014.

Appendices

List of Appendices

| | | |
|----------|--------------------------|-----------|
| A | Content of DVD | 52 |
| B | Personas | 53 |
| C | Wireframes | 56 |
| D | Usability testing | 59 |
| E | Final feedback | 63 |

Appendix A

Content of DVD

- `/src/`
 - `/app-release.apk` - installation archive of AudioCast
 - `/Android-AudioCast/` - Android Studio project
- `/thesis/`
 - `/projekt.pdf` - PDF file of this document
 - `/poster.pdf` - poster for this thesis
 - `/video.mp4` - video for this thesis
 - `/scr/` - source files of this document
 - `/usability-tests/` - recordings from usability tests

Appendix B

Personas

| NAME | BEHAVIOR |
|---|---|
| David "I find myself the perfect example of a lazy student" Student finding the best ways how to work the least | Finds lectures or audiobooks on internet (YouTube) Downloads videos, extracts audio, increases speed of the whole file Plays the audio files using standard music players; Before sleep The process requires various software and many steps |
| DEMOGRAPHICS | NEEDS AND GOALS |
| 25 years old Single Student, IT | Playing local files (audio-books) Adding YouTube videos easily and having them downloaded Increasing speed of playback, sleep timer Having the application for free |

Table B.1: Persona 1

| | |
|--|---|
| <p>NAME</p> <p>Tomas „Podcasts are perfect way how to spend time when traveling.“ Active student, involved in startups</p> | <p>BEHAVIOR</p> <p>Lists through most popular podcasts and selects episodes to download Listens to them continuously while traveling (train, bus, tram) Praises podcast application on iOS with various podcast rankings Wants to listen only the best episodes Forgets progress in listening</p> |
| <p>DEMOGRAPHICS</p> <p>21 years old Single Student, IT, Entrepreneur</p> | <p>NEEDS AND GOALS</p> <p>Finding most popular feeds Online streaming Queuing of „Play later“ Remembering playback position in multiple tracks</p> |

Table B.2: Persona 2

| | |
|--|--|
| <p>NAME</p> <p>Martin „No time to waste my time“ Active manager, learns everything about working with people</p> | <p>BEHAVIOR</p> <p>Lists his subscriptions, manually selects episodes to download Exports best episodes to mp3 and saves for later Listens to locally downloaded mp3 audio-books Uses outdated application because simple usage Plays faster speed everytime</p> |
| <p>DEMOGRAPHICS</p> <p>50 years old Married Employed, HR manager</p> | <p>NEEDS AND GOALS</p> <p>Playing local files (audio-books) Grouping of podcasts by topic/tags Simple interface as possible Playback speed control</p> |

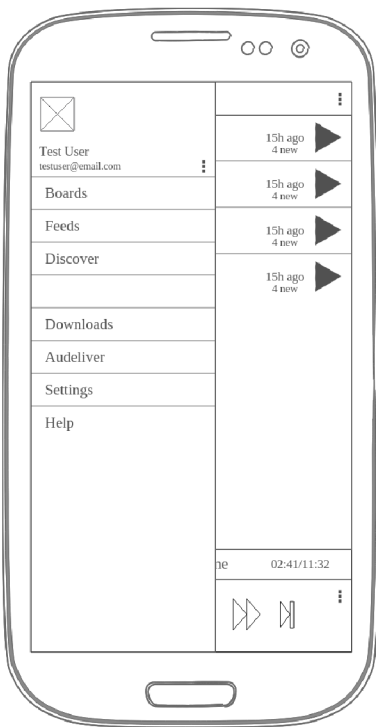
Table B.3: Persona 3

| | |
|---|---|
| <p>NAME</p> <p>Lucie „I listen to music while traveling, studying, training... actually all day long“ Student with headphones on all the time.</p> | <p>BEHAVIOR</p> <p>Subscribes to many SoundClouds RSS feeds using her podcast application Downloads single sets (long episodes) and listens Often listens music before sleep</p> <p>Lacks music which is published only on YouTube</p> |
| <p>DEMOGRAPHICS</p> <p>21 years old Single Student, Electrical engineering Travels a lot</p> | <p>NEEDS AND GOALS</p> <p>Online streaming Organizing subscriptions to groups - Music, Learning, etc. Sleep timer functionality</p> |

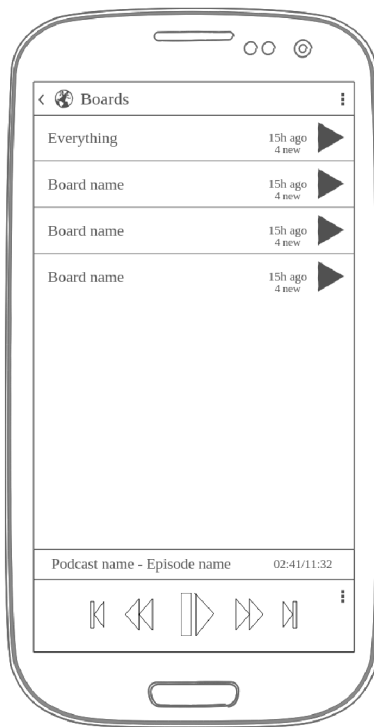
Table B.4: Persona 4

Appendix C

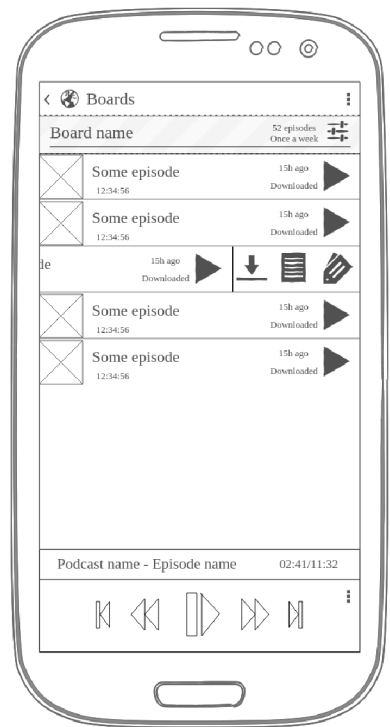
Wireframes



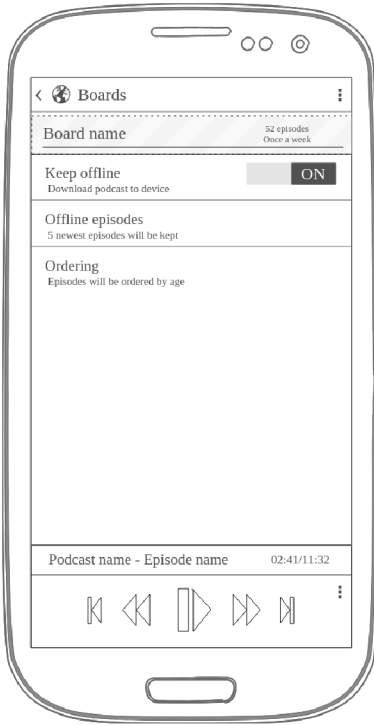
(a) Drawer panel



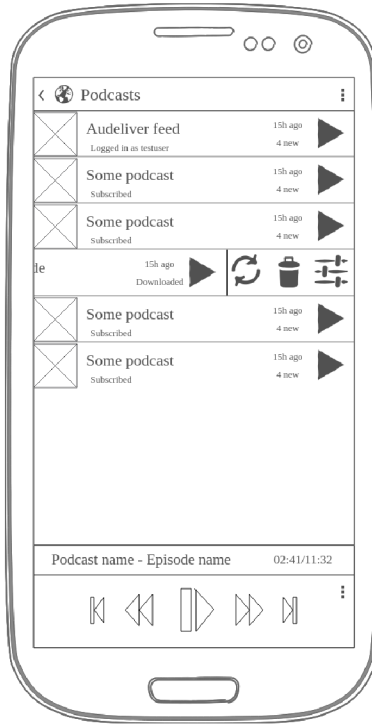
(b) Board list



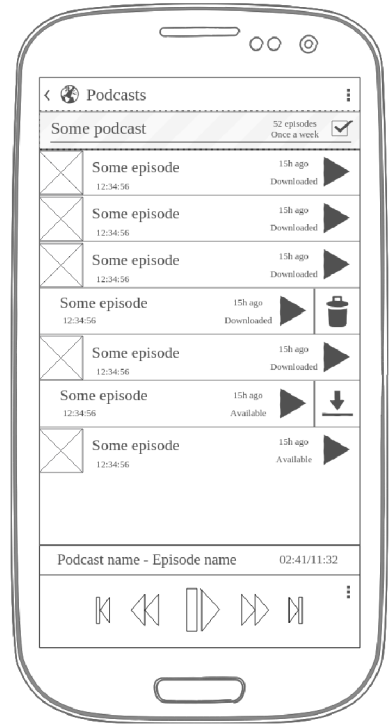
(c) Board detail



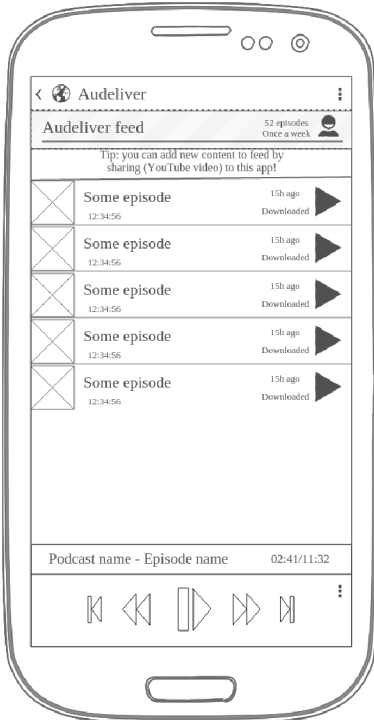
(d) Board preferences



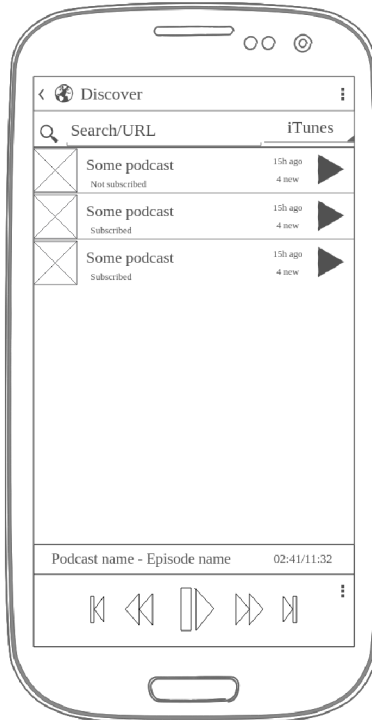
(e) Podcast list



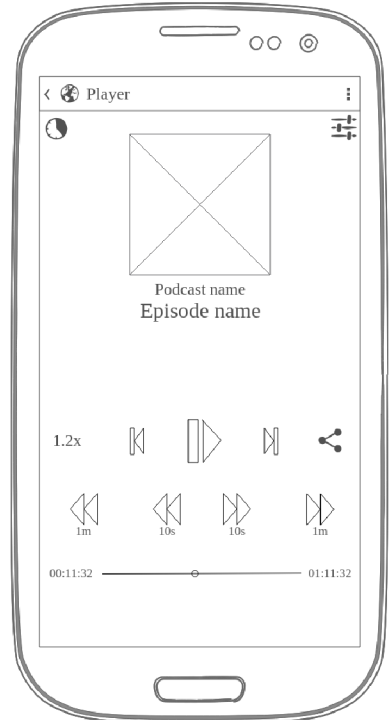
(f) Podcast detail



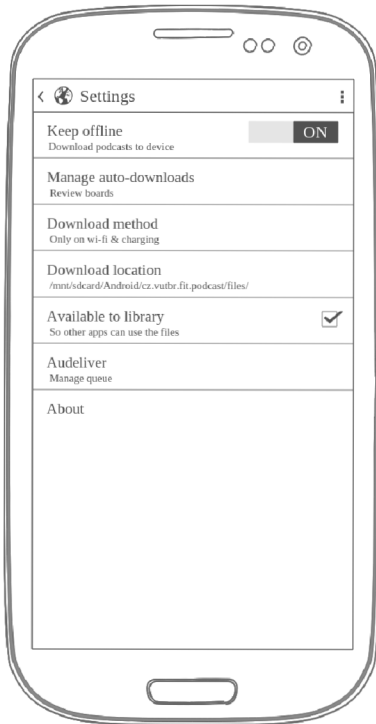
(g) Audeliver feed



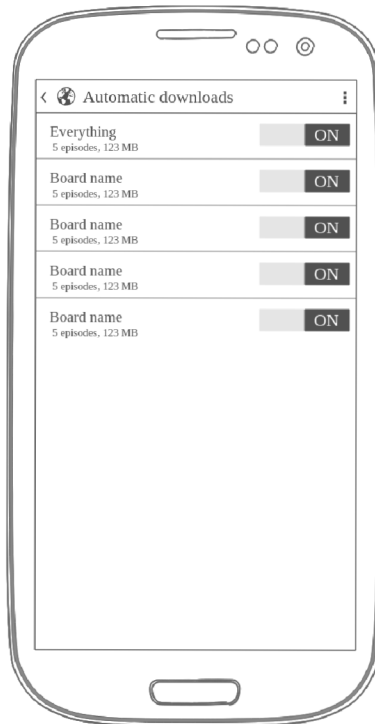
(h) Discover



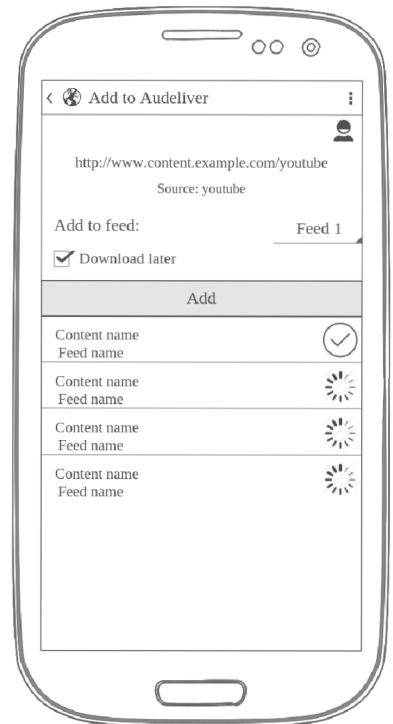
(i) Full player



(j) Main settings



(k) Downloads manager



(l) „Share to“ screen



(m) Downloads queue

Appendix D

Usability testing

Pavel, 12.04.2016

This test was conducted after an option to download items manually was removed. It was an experiment to improve usage of Play Later function and automatic downloads. Before it, users mainly downloaded episodes only manually and played them from Downloads screen. Desired behavior was to move the episodes to Play Later queue where all episodes will be downloaded during periodical automatic synchronization or with forced synchronization. An interactive tutorial was implemented to make users familiar with this feature. It was shown with the first start of application. The tutorial was also tested in this test.

Pavel is a power-user. He regularly listens to podcasts and audio-books using his smart-phone. He is in his early thirties and works as a web-developer. He is also versed in UX and product development.

- Find your favorite podcast, find some interesting episodes and download them for later.

Is Discover function easy to find and explanatory?

I expected it to be in the top-right part of the screen, in the top bar. But when I saw the button [floating action button] it was clear to me.

Would you use more details about the episodes?

It is enough. The date of publication is the most important for me.

Is the Play Later explained in tutorial well? Did you understand what is it good for?

I understood the concept but I did not like it. A button to download the episode directly would be much better. The way with Play Later requires a lot of clicks. During the download I did not see any feedback in the application. Only the notifications showed the progress. It was confusing.

Is manual download intuitive?

The button to open the screen was the only option so yes. However the screen for automatic downloads was confusing. It took me a lot of time to understand what does the first entry mean. Also, when downloading episodes manually I would like to select number of episodes to download.

- Create Audeliver.com account and find your podcast.

Is the login button intuitive?

Not really. I was searching between the text fields in drawer. If you didn't point me out I probably wouldn't find it.

Is register/login process smooth and easy?

There are no misleading options so yes. I didn't like that I had to open web-page and register there. Also the need to enter my email and password after it once again wasn't good.

Where would you expect your Audeliver.com podcasts?

Just where it was. In the list of podcasts.

- Change playback speed of the currently played episode.

Is this control accessible? Where would you rather put it?

Yes, I found it in the Player. It was obvious and worked great.

- Find a YouTube video and add it to your Audeliver.com podcast.

Where from did you know about this feature?

I don't know. I didn't really read the tutorial, there were too many screens. But I seen it in some applications that I could share a song to some other application.

Would you welcome more info about the episode?

Yes. I seen only the title and link. A picture from the video would be very nice.

- Import your collection of audio-books and listen to one.

Did you understand the two different add-modes?

Yes. I use similar application which has this system too. But it is well described here too.

What features would you want in the file-chooser?

None. It does what it should do. It is just fine.

Jakub, 19.04.2016

- Find your favorite podcast, find some interesting episodes and download them for later.

Is Discover function easy to find and explanatory?

Yes, I found it. But I think that open drawer on startup would be very useful. Oh, and the button in podcasts list is great, I wouldn't search for it in the top bar.

Would you use more details about the episodes?

Yes, it is absolutely enough.

Is the Play Later explained in tutorial well? Did you understand what is it good for?

It is self explanatory, but I wouldn't use it because I rather listen to a single podcast.

Is manual download intuitive?

Yes, I understood it. I didn't like the dialog when I wanted to download it manually. I think that selecting number of episodes directly would be better.

- Create Audeliver.com account and find your podcast.

Is the login button intuitive?

Round text above login button with LOGIN/AUDELIVER

Is register/login process smooth and easy?

Yes, no problems at all.

Where would you expect your Audeliver.com podcasts

In the list of podcasts, as they are.

- Change playback speed of the currently played episode.

Is this control accessible? Where would you rather put it?

It is on a good place. But I didn't like the picker. Slider would be better.

- Find a YouTube video and add it to your Audeliver.com podcast.

Where from did you know about this feature?

I wanted to use the QR code at first, it was more intuitive. But then I read it in the tutorial. When I used the QR scanner I didn't like that I need to confirm the code. It could take the first code it detects.

Would you welcome more info about the episode?

It's just fine. I was guided by the image which is the same as in YouTube application.

- Import your collection of audio-books and listen to one.

Did you understand the two different add-modes?

I didn't use audio-books before so I thought that one audio file was one audio book. So I didn't understand the single or multiple folder picker. But when I saw how a audio-book is saved, it was intuitive.

Riccardo, 18.05.2016

- Find your favorite podcast, subscribe to it, find some interesting episodes and play them.

Is Discover function easy to find and explanatory? Yes, it was obvious after I saw it in the panel. It took me a while because I was getting familiar with the application. But it is on the right spot. The search screen is clear and intuitive.

Would you use more details about the episodes? No, it has everything I need. Description and date of publication.

Is the Play Later feature explained in tutorial well? Did you understand what is it good for? Yes, it is for saving episodes for later and application will download them.

- Set your favorite podcast to automatically download 4 newest episodes. Then download two latest episodes from the podcast.

Did you find the automatic downloads easily? Was it intuitive? Yes, it was the only button. However, it could be bigger. But the screen was really confusing for me. The button to download indicates the automatic download for me. I didn't expect that I need to click on the item to change it.

Do you know about more ways how to download latest episodes? Well. Using the download in automatic download or manually I guess.

- Create an Audeliver account and find your podcast.

Is the login button intuitive? Yes, no problems. The orange text attracted my eye.

Is register/login process smooth and easy? Yes, no problem.

Where would you expect your Audeliver podcasts to be? In the podcast list. On the top, as it is. However, it can get bad if you have a lot of Audeliver podcasts. In the case it would be good to have separate screen for it.

- Change the playback speed of the currently played episode.

Is this control accessible? Where would you rather put it? Yes, I saw it when I was going through the application at the beginning.

- Find a YouTube video on your mobile and add it to your Audeliver podcast.

Did you know about this feature? Where from? From the tutorial I guess.

Would you welcome more info about the episode? No, it has what it should have.

- Find a YouTube video on your computer and add it to your Audeliver podcast.

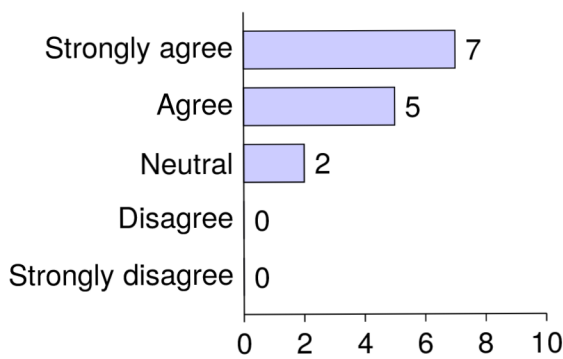
Did you know about this feature? Where from? Yes, I saw the QR scanner in the search screen. But I didn't think of generating a QR code from YouTube URL.

- Import your collection of audio-books and listen to one.

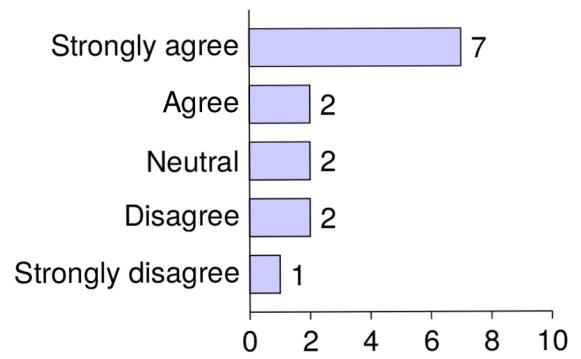
How did you understand the two different add-modes? One is for a single folder, another for folder with more audio-books. However, I didn't like the text. It is just a detail, but I would make the texts on the same size.

Appendix E

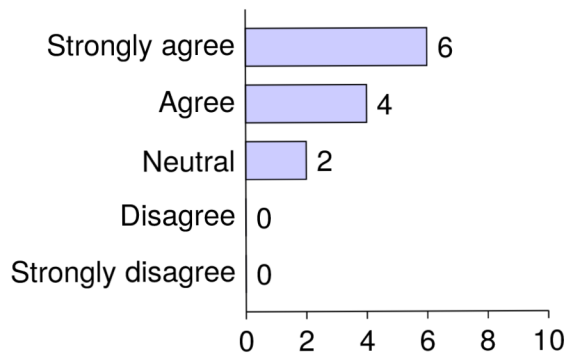
Final feedback



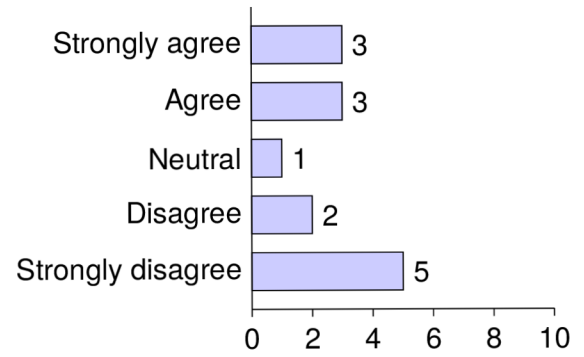
(a) The controls of the playback in application and in notification are intuitive.



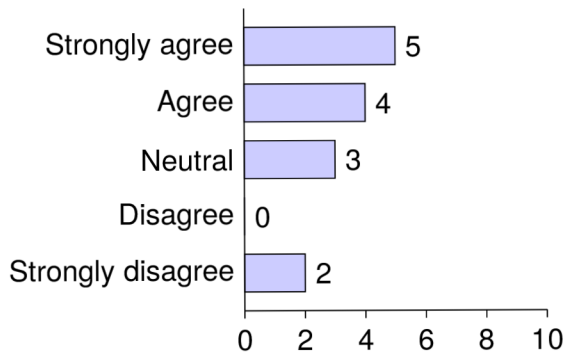
(b) The player screen is explanatory and has all important controls.



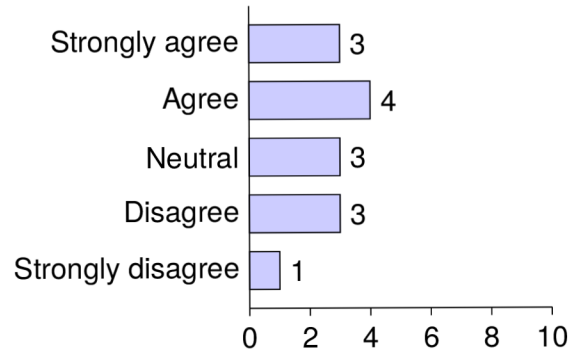
(a) The podcast detail contains all important information.



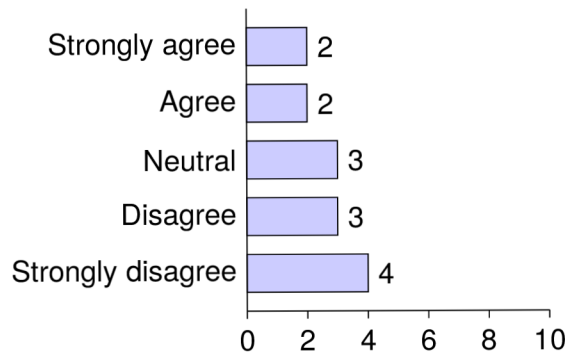
(b) The tutorial is useful and I read it.



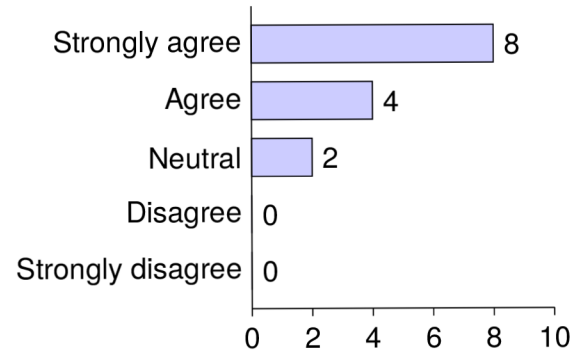
(a) I understand what is a podcast.



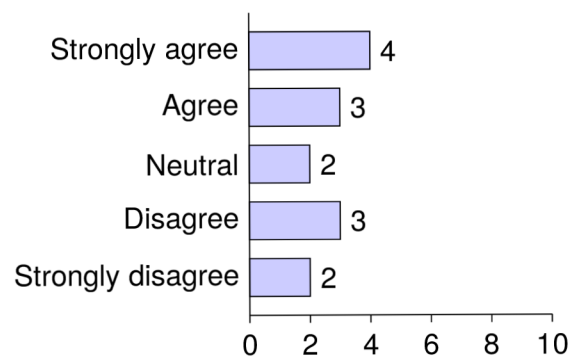
(b) Settings screen is clear and explanatory.



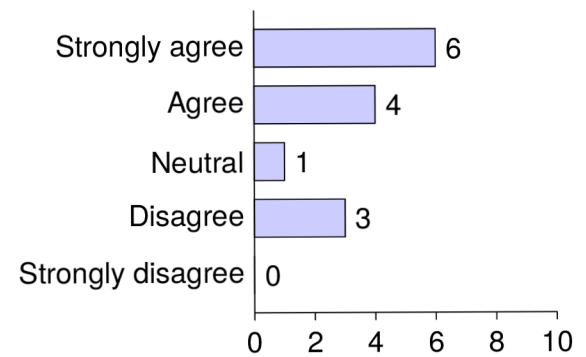
(a) Automatic-downloads screen is intuitive.



(b) Register/login to Audeliver is easy and smooth.



(a) Adding content to Audeliver account is easy and smooth.



(b) Discovery screen is intuitive and easy to use.