UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Fredi Arro

KaiOS Application For Listening Podcasts

Bachelor's Thesis (9 ECTS)

Supervisor: Ljubov Jaanuska

Tartu 2020

KaiOS Application For Podcast Listening

Abstract:

The aim of the present thesis was to develop an application for the KaiOS devices for listening podcasts. No convenient alternative application had been created before. In this thesis, the waterfall model was used for the development process. The requirements, design and structure were implemented using HTML, CSS, and jQuery. iTunes Search API was used for searching new podcast. IndexedDB database stored the data about the subscribed podcasts. In the future, it is planned to upload the application into KaiStore and add new features such as push notifications and download episodes.

Keywords:

KaiOS, web-application, podcast, HTML, CSS, JavaScript, jQuery, IndexedDB, iTunes Search API, Waterfall model

CERCS: P175 Informatics

KaiOS rakenduse taskuhäälingute kuulamiseks

Lühikokkuvõte:

Lõputöö tulemusena loodi rakendus, mis võimaldab kuulata taskuhäälinguid KaiOS seadmetega. Rakenduse abil on kasutajal võimalus otsida, tellida ja kuulata taskuhäälinguid. Selle töö kirjutamise hetkel ei olnud ühtegi mugavat võimalust taskuhäälingute kuulamiseks KaiOS seadmetega. Arendusprotsess toimus koskmudeli alusel. Esmalt pandi paika nõudmised ja piirangud, mida rakendus peab täitma. Järgnevalt kujundati rakenduse kasutajaliides ja üldine struktuur. Viimasena realiseerti funktionaalsused, loodud disain ja struktuur kasutades järgnevaid tehnoloogiaid: HTML, CSS ja jQuery. Uute taskuhäälingute otsimiseks kasutati iTunes otsimise programmiliidest (*ingl.k iTunes Search API*). Andmete talletamiseks kasutati IndexedDB andmebaasi. Edasisise arendusega on plaanis sisse viia tõuketeated ja võimalus episoode allalaadida. Lisaks laetakse rakendus tulevikus KaiStore üles.

Võtmesõnad:

KaiOS, veebirakendus, taskuhääling, HTML, CSS, HTML, CSS, JavaScript, jQuery, IndexedDB, iTunes Search API, koskmudel

CERCS: P175 Informaatika

Table of Contents

1. Introduction

In today's world, where almost everything is online, users like to consume media when it is possible and comfortable for them. The number of people listening to audio online is growing [1]. This also affects podcasts, which are also rising in popularity, according to Edison Research [1]. The podcast format also gives the user the possibility to listen, whenever the user likes to. In addition to that, most podcasts are free.

As Jakob Rosin has written [2], podcasts are like prerecorded radio shows. Unlike the radio shows, almost anyone, who has a microphone and the possibility to record, can start a podcast. Podcasts are not broadcasted like the radio shows. Instead, podcasts are uploaded to the internet. Every time the creator uploads an episode of a podcast, it is delivered to the listener's phone via an application. The user can listen to the episode any time.

Users can use a computer or smartphone to listen to podcasts. On both devices the user can use a browser for listening. Since a browser is not very comfortable for this task, there are applications for podcast listening. For a computers, the most known program of this kind is Spotify. The same application exists for smartphones, but there is more variety of them. Jakob Rosin, in the year 2018 [2], highlighted the applications Overcast and Pocket Casts.

In contrast with the smartphones, some people have chosen to use feature phones[1]. Most of these feature phones run an operating system named KaiOS. According to statcounter [3], the third most used mobile operating system in the last 12 months (from March of 2019 to March 2020). For the users, it is almost impossible to listen to podcasts on their phones. People have to download the episodes manually to their computer, and then transfer them to the phone. This is an inconvenient solution, which decreases the flexibility of the podcast format. By the time of April of 2020, there had been no other solution known to the author.

The goal of this thesis, is to develop an application for listening to podcasts, on KaiOS phones. The application has to facilitate the user:

- listening to a podcast episode;
- subscription to the podcast.

The thesis is divided into nine chapters. The first chapter is the introduction. The second topic gives an overview of the operating system and its popularity. In addition to that, the chosen development process model is described in detail. In the next chapter, the limitations, non-functional, and functional requirements are described. At the end of the chapter there are use cases. The design and navigation of the application is covered in the fourth chapter. In addition to that, the architecture for data is defined. The fifth chapter addresses the implementation process. In sixth chapter, the future potential of the application is detailed. The thesis and its results are summarized in the seventh chapter.

[1] Feature phone is a phone with minimum amount of features. The main functionalities include calling and sending text-messages. The phone usually has buttons to navigate the user interface.

2. Background

The podcast application is developed for the KaiOS. The first half of this chapter gives an overview of the operating system. In the second half of this chapter, the chosen software development process is described.

2.1 KaiOS – Mobile Operating System

KaiOS is a mobile operating system [4]. It is a fork from Boot to Gecko OS which was an open source fork from Firefox OS [5,6]. The source code for KaiOS is available on their GitHub, but the parts of the code which are owned by their partners is not on there.

KaiOS is used on feature phones like Nokia 8810 4G [7]. KaiOS phones usually have non-touch screens and use buttons for navigation [8]. These are cheap phones with a small number of essential features. Before, these type of cheap devices had usually one or two functionalities, like calling and sending text-messages. KaiOS is broadening the features list for these devices. Phones running KaiOS have 3G/4G and WiFi [8], which allow the user to access the web. This makes accessing the internet much cheaper for most people, since the phones running KaiOS are cheaper than their competitors, like Android and iOS devices.

According to KaiOS' website [9], there are currently around 100 million devices worldwide running KaiOS. The absolute value seems to be large; nevertheless, the share of KaiOS devices worldwide was 0.53% in the last 12 months [3]. Since Android and iOS have established themselves so well, it seems like KaiOS does not have a chance against them in the worldwide market share. On the other hand, in India, where people's per capita income, about 11500 Indian rupees [10] (about 140 euros), is much lower compared to the the worlds average per capita income, about 16000 euros [11], the market shares are different. According to *statcounter[12,13]*, the share of KaiOS devices in India has risen. From the March of 2018 to March 2019, the market share of KaiOS devices was 2.2%. It was in fourth, behind Android (89.7%), iOS (2.9%), and unknown (2.7%). In the last 12 months (from March 2019 to March 2020) KaiOS devices' market share was 3.1%. It had become the second largest mobile operating system in India, just ahead of iOS (2.7%) and a behind Android (92.9%). This growth over the last two years showed, that in markets like India, this is had a lot of potential.

The next descriptions of KaiOS layers is referenced from KaiOS' developer portal [14]. KaiOS' architecture has three layers (Figure 1) [14].

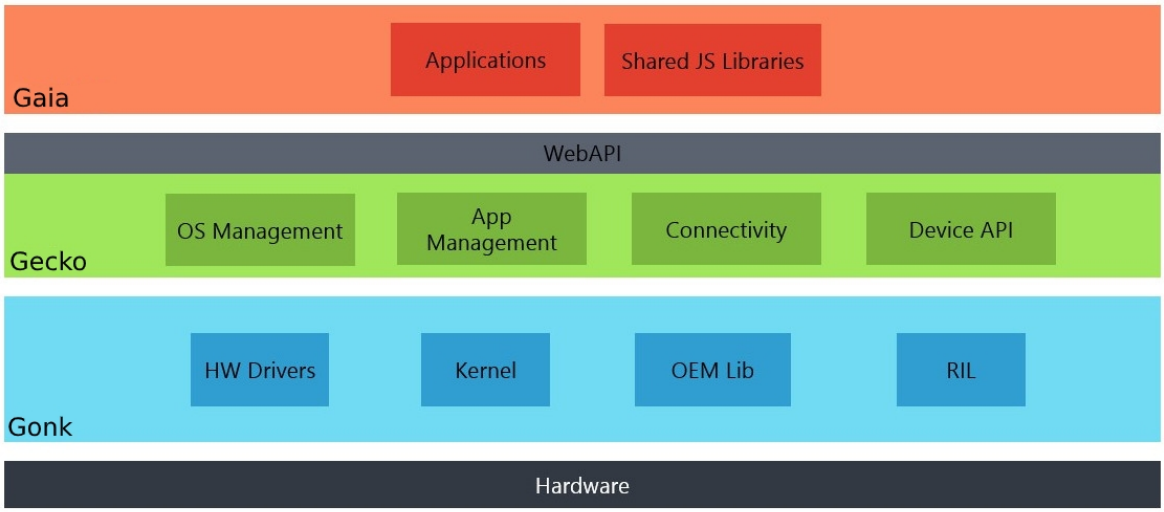

Figure 1: KaiOS layers [12]

Since Android and iOS have established themselves so well, it seems like KaiOS does not have a chance against them in the worldwide market share.

Gaia is the front-end of KaiOS. It renders the graphics used in the operating system, such as the lock-screen and the menu. Gaia uses Web API to communicate with rest of the operating system and hardware, for example buttons. The third party applications can be installed next to the Gaia layer.

The Gecko layer creates the run-time for the KaiOS applications. It supports HTML, CSS, and JavaScript. Gecko has multiple purposes, the most notable are:

- it parses and renders HTML5;
- connects HTML to hardware.

Gonk is described in the KaiOS development portal as the lower level operating system. It consists of a Linux kernel and a hardware abstraction layer. Gonk is the link between Gecko and phone's hardware.

KaiOS applications are web-applications, which means they use HTML, CSS, and JavaScript. The built app is run by Gecko run-time [15]. This makes developing applications for KaiOS accessible and easy.

2.2 Waterfall Model – The Linear Process

Ivan Marsic states [16], that software development has five phases:

1. requirements specification;
2. design;
3. implementation;
4. testing;
5. operation and maintenance.

These phases can be covered in different ways, which make the different models of software development. In the beginning, the inspiration for software development models came from other engineering disciplines [16]. The waterfall model method is an example of that. Waterfall model is linear. The linearity means, that the developer has to finish with the current step to move on to next the one (see Fig. 2).

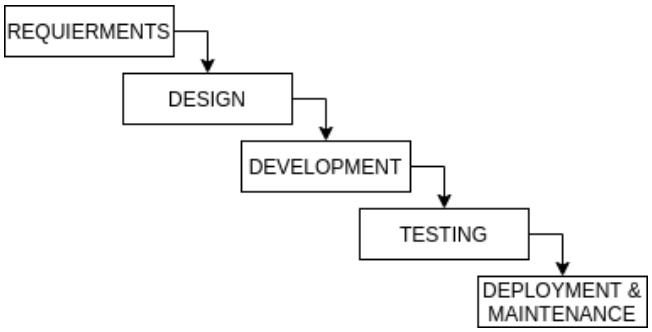

Figure 2. Waterfall model described by Ivan Marsic [16]

For this thesis, the author chose the waterfall model since [17]:

- it is understandable and easy to implement;
- it is suitable for inexperienced teams and developers;
- it is linearity makes it easy to implement.;
- the model is used when requirements are very clear and do not change during the development.

The development process section will cover the following phases of the waterfall process [16]:

- requirements specification;
- design;
- implementation;

This thesis does not cover the testing and the deployment and maintenance phases. The testing and the deployment of application is done when it is submitted to the KaiStore [18].

3. Requirements Specification

The requirements to the application development were set taking into account the KaiOS devices' performance capabilities and hardware. Also some limitations were set by KaiStore submission [18]. The author gathered the functional requirements from using the Android and web-application Castbox [19]. While gathering the requirements, the limitations of KaiOS devices were kept in mind, to make the implementation achievable.

3.1 Limitations

The limitations set to the applications by the KaiOS devices are as follows:

1. screen size is 240 pixels by 320 pixels or the other way around, depending on the phone;
2. all the input and navigating has to be done by buttons;
3. the application has to be able to run with 512 megabytes of RAM;
4. application has to follow KaiOS user interface design guidelines;
5. the application must have an icon.

3.2 Non-functional Requirements

Next, a list of the non-functional requirements for the application are set:

1. the application has to be compatible with portrait or landscape screens;
2. the application has to store the information about subscribed podcasts in the phone, using a local database;
3. the application has to work with Mozilla Firefox version 59 or older [20].

3.3 Functional Requirements

The functional requirement to the application are as follows:

1. the user can search for new podcasts, that he/she has not subscribed to;
2. the user can subscribe to a podcast;
3. the user can unsubscribe from podcast;
4. the user can see a list of podcasts, that he/she has subscribe to;
5. the user can search for podcasts, that he/she has subscribed to;
6. the user can listen to an episode of a podcast;
7. while listening, the user can move to any timestamp on the podcast timeline;
8. the user can add an episode to a listening queue;
9. the user can reverse the order of episodes.

3.4 Use Cases

In the following section, various use cases are described. The user stories demonstrate how the stated functional requirements can be implemented in the real application usage.

UC-01 Subscribe to a podcast

Requirements: 1, 2

Preconditions: the application has been opened and user is on the “Search” tab

Result: the user subscribes to a podcast

Main scenario:

1. the user enter the name of the podcast they want to find;
2. hits “SEARCH”;
3. the user moves to the podcast;
4. the user hits “Subscribe” button.

An alternative scenario 1:

1. the user enter the name of the podcast he/she wants to find;
2. the user hits “SEARCH”;
3. user’s podcast does not show up;
4. a message is shown that no podcast matches that search.

An alternative scenario 2:

1. the user enter the name of the podcast he/she wants to find but is already subscribed to;
2. the user hits “SEARCH”;
3. the user moves to the podcast;
4. since the user cannot subscribe to the podcast twice, the “Unsubscribe” button is displayed.

UC-02 Listen to an episode from a podcast

Requirements: 4, 5, 6

Preconditions: the application has been opened and the user is on the “Search” tab.

Result: the user listens to an episode.

Main scenario:

1. the user moves to the “Subscriptions” tab;
2. the user scrolls down to his preferred podcast;
3. the user opens the podcast;
4. the user selects an episode;
5. the user hits the “PLAY” button.

An alternative scenario 1:

1. the user moves to the “Subscriptions” tab;
2. the user searches for their preferred podcast;
3. the user hits “SEARCH”;
4. the podcast shows up;

5. the user opens the podcast;
6. the user selects an episode;
7. the user hits the “PLAY” button.

An alternative scenario 2:

1. the user searches for their preferred podcast;
2. the user hits the “SEARCH”;
3. the podcast shows up;
4. the user opens the podcast;
5. the user selects an episode;
6. the user hits the “PLAY” button.

UC-03 Listen to a specific moment in an episode

Requirements: 6, 7

Preconditions: an episode is playing and the user is in the “Search” tab.

Result: A specific moment can be heard from an episode.

Main scenario:

1. the user hits “Player” button;
2. the player opens up;
3. the user hits “Move to” button;
4. “Move to” options with timestamp selection opens up;
5. the user selects the specific timestamp and hits “SELECT”.

UC-04 Add an episode to the queue

Requirements: 8

Preconditions: User is in the “Subscriptions” tab.

Result: An episode of a podcast is added to the queue.

Main scenario:

1. the user finds a podcast from the “Subscriptions” tab;
2. the podcast is opened;
3. the user finds an episode they want to add to the queue;
4. the user clicks “Options”;
5. the user clicks “Add to queue”;
6. the popup confirms, that the episode has been added to the queue.

An alternative scenario :

1. the user moves to the “Search” tab;
2. the user searches for his/hers preferred podcast;
3. the podcast is found and opened;

4. the user finds an episode they want to add to the queue;
5. the user clicks “Options”;
6. the user clicks “Add to queue”;
7. the popup confirms, that the episode has been added to the queue.

UC-05 Start listening a podcast from the first episode

Requirements: 6, 9

Preconditions: the user is in “Subscriptions” tab.

Result: the episodes are displayed in the reverse order.

Main scenario:

1. the user finds a preferred podcasts;
2. the podcast is opened;
3. the user hits “Options”;
4. the user hits “Ascending/descending”;
5. the podcast episodes are shown in an ascending order (first episode is on top);
6. the user clicks “PLAY” on first episode.

An alternative scenario:

1. the user finds a preferred podcasts;
2. the podcast is opened;
3. the user scrolls down to the first episode;
4. the user clicks “PLAY” on first episode.

UC-06 Unsubscribe from a podcast

Requirements: 2, 3, 4, 5

Preconditions: the application has been opened and the user is on the “Subscriptions” tab.

Result: the user is successfully unsubscribed from the podcast

Main scenario:

1. the user finds the podcasts;
2. the user hits “Unsubscribe”;
3. popup confirms the canceling of the subscription.

An alternative scenario:

4. the user moves to “Search” tab;
5. the user finds the podcasts;
6. the user hits “Unsubscribe”;
7. the popup confirms the canceling of the subscription.

4. Application Design

The application design does not mean the interface design only. According to Inga Petuhhov [21], the application design also includes planning the navigation between different front-end elements, data storage structure, and the over-all architecture of the application. In the present chapter, these topics are covered.

4.1 Design of the User Interface

A good user interface is the one that the user does not notice. It is intuitive for the user. On the feature phones, the user notices every inconvenience because the interaction with the phone is limited to buttons. Also the limited hardware resource may cause stuttering, which means the design has to be lightweight. In addition to the lightness and smoothness, the e screen size of the devices has to be considered.

KaiOS has a helpful design guide [22] which describes how to make user interface more user-friendly. The overall idea of KaiOS' design is to keep it simple. There should not be too much information on the screen, so it does not get difficult for the user to follow. Also all the elements have to be easy to see since the screen size of most devises is between 2.4 and 2.8 inches [8].

In the design guide, specifications of the most common elements used in KaiOS can be found [23]. The following elements from the guidelines were used by the author: header, tab, software key, list, input, options menu, value selector, toast. All the other elements were created and designed by the author.

To satisfy all the functional requirements of the application, five views are needed. These are subscription, search, podcast, player, and queue. Next, the views and the design are described. The mock-ups are drawn using Pencil software [9].

Subscription and Search Views

When the user opens the application, they are directed to the subscription view (see Fig. 8 in the Appendix I). The view shows all the podcasts that the user is subscribed to. The result is displayed in the list format. Due to the list, it is easy to search and navigate among the podcasts. The search view (see Fig. 9 in the Appendix I) has the same layout as the subscription view; however, the search view shows only the podcasts that the user has been looking for. The subscription and search views are displayed in a tab view [22], which can be navigated with the directional pad. The views have been developed using the design guide [20] and following the guidelines.

Podcast and Queue Views

The podcast and queue view (see Fig. 10 and Fig 11. in the Appendix I) are very similar to each other. Both of them display a list of episodes. In the podcast view, the user can see and search the episodes of a certain podcast. In the queue view, the user can see all the episodes that have been added to the queue. In the queue view, there is no search functionality. These views have been developed following the guidelines given in [22].

Player View

The player view (see Fig. 12 in the Appendix I) shows essential information about the playing episode. This view has been designed by the author since there are no specifications in [22] for some of the elements used in the view. The view occupies about

a half of the screen, it has a contrasting background. Due to the latter, the view is easy to notice and read.

4.2 Navigation and Structure of Screens

The structure of the application had to be simple for the user to use. This meant, that all the options to navigate through different screens had to be labeled or already known from previously using KaiOS. For example, hitting the back button to go back to the previous screen, is known to users, because it is used in the KaiOS menus.

In the following Figure 3, there is displayed the structure and navigation of the application.

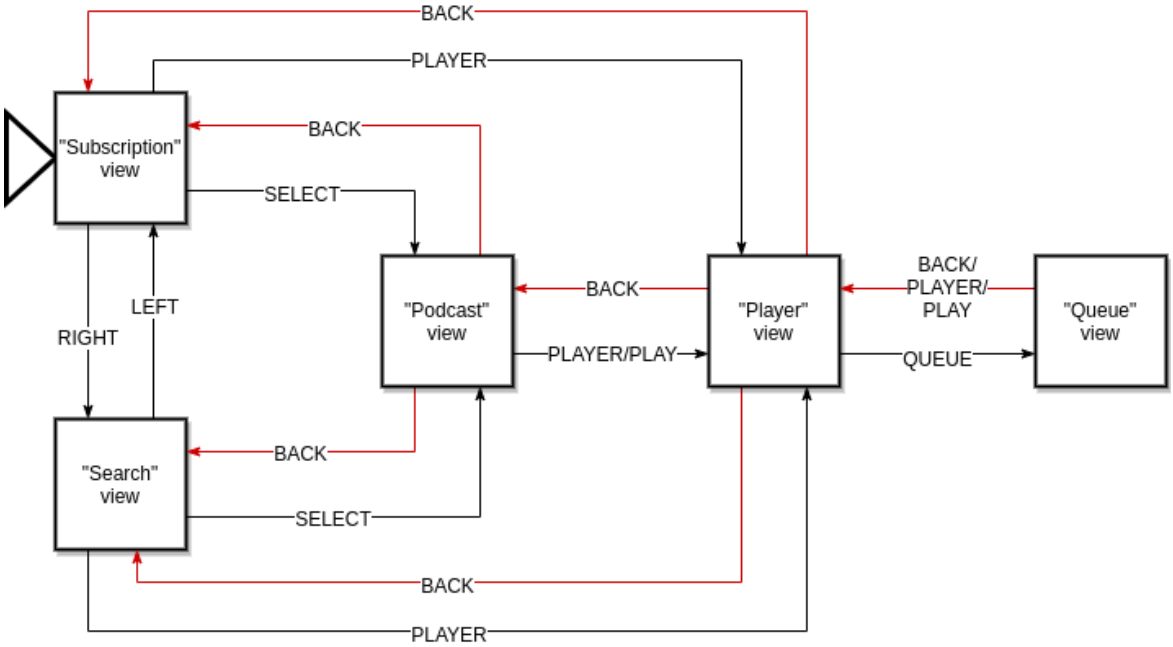

Figure 3: Application navigation structure

All the labels, except “BACK”, for the arrows are shown on the soft keys section, throughout the application. The screens, where there are multiple arrows with the label “BACK”, the user is routed to the screen, which they were on previously.

4.3 Data Storage Design

The application would be used to listen and subscribe to podcasts. For listening, the audio will be streamed from the internet, so there was no need to handle downloaded episodes. All the information about the subscribed podcasts was planned to be stored in a database. The database would be local, so there would not be a need for a server. This keeps the development process simple. In addition to that, it saves the device’s resources, since it would not have to ask for the data from the internet every time it needs it.

In the database, there had to be tables for the following information:

- subscribed podcasts’ information;
- episodes of the subscribed podcasts;
- queue.

The first table is updated, when a user subscribes to new podcast. When this happens, a table for the podcast’s episodes is created. The episodes tables are updated, when the

application is starting up. The queue table is updated, when the user adds something to the queue.

4.4 Structure of the Application

The Figure 4. shows the overall planned structure of the application. In the figure, there are show different components of the application and why they connect to each other.

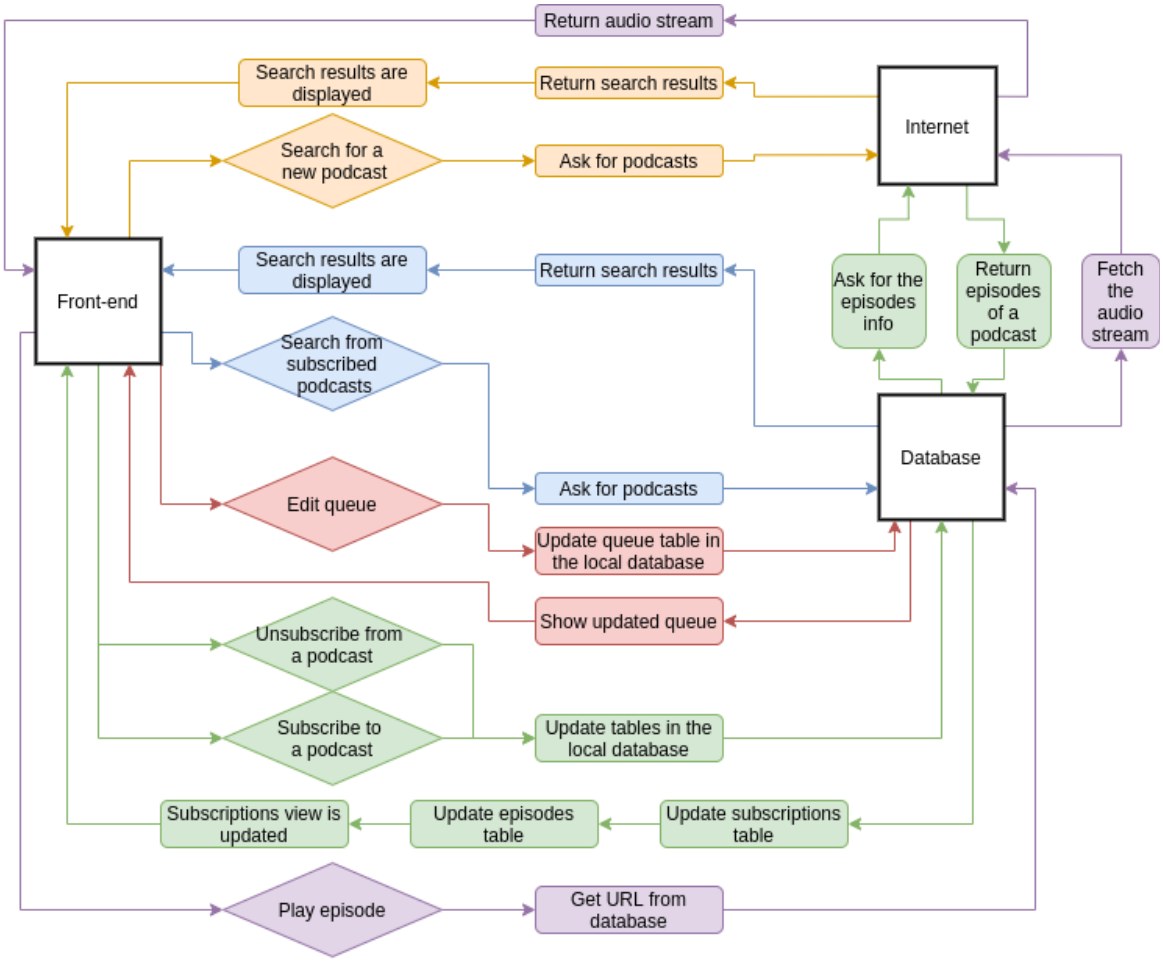

Figure 4: Structure of the application

The front-end connects to the internet, when the user tries to search for a new podcast. When the user want's to subscribe to a podcast, the front-end connects to the database, which fetches data from the internet. Finally the updated database is returned to front-end. When an episode is added or removed from the queue, the front-end connects only to the database. Finally, when a user wants to listen to the episode, the front-end connects to the database. The database then asks for information from the internet, which is finally returned to the front-end.

5. Implementation

This chapter describes the development process of the application. This section will cover The development tools, structure and design of front-end and back-end of the application are covered in detail.

5.1 Development Tools

The development tools were chosen following recommendations given in the KaiOS Developer Portal [25].

WebStorm

WebStorm is an IDE developed by JetBrains [26]. It supports most programming and markup languages needed for the web-development such as JavaScript, TypeScript, CSS. WebStorm helps manage files and directories related to the project. For this thesis, the version 2019.2.3 has been used.

Git and GitHub

To manage the versions of the application, Git (version 2.17.1) has been used in combination with GitHub. Git is a free and open source distributed version control system to handle everything from small to very large projects with speed and efficiency [27].

GitHub is a git-based collaboration and sharing service that also provides hosting [28].

KaiOS Run-time Application

KaiOS Run-time application is a simulator provided by KaiOS. The simulator launches Gaia and web-applications in Gecko based environment which is similar to an actual device [29]. This application has been used to test the design and the interaction with the button. The version 2.5 has been used in this thesis.

Nokia 8810 4G

Nokia 8810 4G is a feature phone with KaiOS. It has 512MB of RAM and a portrait screen [8]. This device has been used to test the application.

5.2 Used Technologies

HTML

Hypertext Markup Language [30], also known as HTML, is the standard markup language for websites. In this thesis, HTML5 has been used.

CSS

CSS (Cascading Style Sheets [31]) has been used alongside with HTML to style the HTML document. In this thesis, CSS3 has been used. It made styling of the document much easier since multiple pages and elements could be controlled from one file [32].

KaiUI

KaiUI is a public repository created by Jahidur Rahman Nadim [33]. It has most of the elements from the KaiOS UI elements guidelines (e.g. header, separator, software key, toast, list, options menu, button, input, table, and progress). All the elements are styled with CSS, which keeps the solution lightweight. It is worth to note that the KaiUI

repository is not the same as the one mentioned in the KaiOS user interface guidelines [22], but both of them follow the guidelines.

jQuery

jQuery is a fast, small, and feature-rich JavaScript library [34]. It was released by John Resig in the year 2006 [35]. jQuery helps bundle up common and redundant tasks into methods that can be called with a single line [36]. The simplicity and a small footprint (it takes up 30 kilobytes of the space [34]) were the reasons why it was chosen for this application . The version 3.4.1 was used in this thesis.

IndexedDB

IndexedDB is a low level API for local storage [37]. It provides a solution to storing large structured data. IndexedDB has been used in the application since it was the only local storage available for KaiOS.

iTunes Search API

The Search API is a tool which allows to search for content in the iTunes Store and Apple Books Store [38]. The API helped keep the application lightweight.

5.3 Application Structure

The application of focus has a structure common to most web-applications. Figure 3 demonstrates how the chosen technologies are tied to each other. As it seen from the

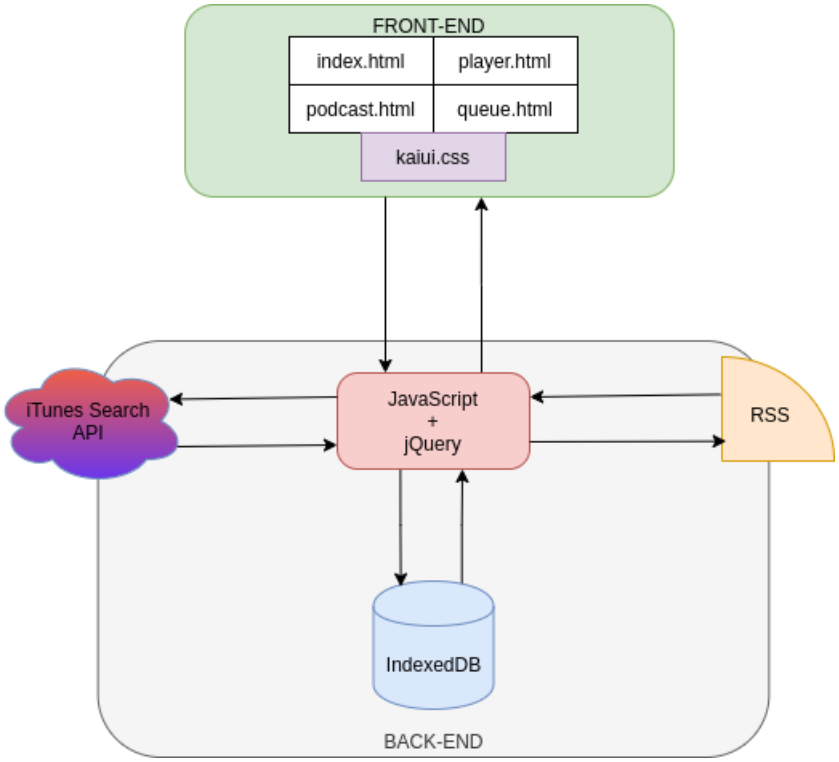

Figure 5: Application Structure

figure, the center-piece of the application is JavaScript with jQuery. The following subsections describe the elements of Figure 3 in detail.

5.3.1 Front-end

The front-end design of the application was implemented with the aid of HTML and CSS. Some of the elements were developed by the author, some of the UI elements were forked from the public GitHub repository KaiUI by Jahidur Rahman Nadim [33].

The author forked the repository. In author's repository the following elements were modified: header, software key, list, options menu and input. Some of the elements were not described in the original CSS file. The author added the styling for tabs and player elements.

The manipulations within views and between them were implemented using JavaScript. For navigation, the lists and tabs were created by the author using the corresponding JavaScript files.

Also JavaScript was used to implement manipulations between different views. This also handled all the pressed key (e.g. "SELECT" and "Player"). In addition to the navigation, JavaScript created the lists of podcasts or episodes.

5.3.2 Back-end

The back-end of the application handles the search of podcasts, subscription to a podcast, browsing episodes of a podcast, and editing the queue. All these actions are tied to the database. The next section gives an overview of the database.

Database Structure

The indexedDB tables, are shown in Figure 6.

The

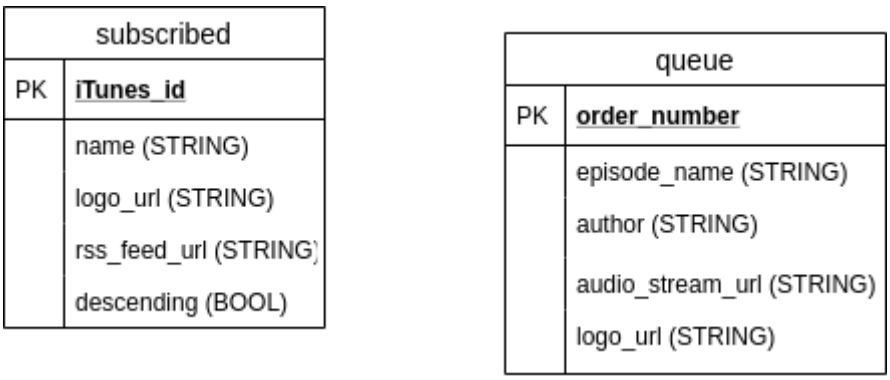

Figure 6: Applications database structure

"subscribed" table keeps all the podcasts that the user has subscribed to. The data of this table are fetched using the iTunes search API.

The *queue* table handles the queue. The *episode_name* field is used to display it in the queue view. The *episode_name*, *author_name*, *audio_stream_url*, and *logo_url* are used to in the player view, when the episode is played.

Handling Search

In the application, the user can search for new podcasts and subscribe to them. The search is implemented using JavaScript and iTunes Search API. When the user's entered term is sent to iTunes Search API, it returns an XML file containing the search results. The XML is processed by JavaScript and the first 20 results are show on the screen. From the XML, RSS feed URL, logo URL, and iTunes saved to the to the HTML elements. HTML division holding a podcast gets an id tag which corresponds to the iTunes id.

If the user wants to see an episode, they have to open the corresponding podcast via the button. Next, the RSS feed is fetched and processed by JavaScript. The processed data is show on the screen.

Handling Subscriptions

Subscription is handled by IndexedDB, RSS and JavaScript. When the user decides to subscribe to a new podcast, the podcast is added to the “subscribed” table with the needed data. When the user decides to unsubscribe from the podcast, the entry is removed from the *subscriptions* table.

All the episode’s tables are updated from the RSS feed every time, when the user opens the certain podcast view..

6. Application Potential

The further development of the application addresses the user experience and the reusability of the code

Downloading episodes

Currently, all the episodes are streamed from the Internet. This method is not very efficient – the application consumes a lot of data and energy. The option to download episodes can eliminate both problems and make the application work offline.

Push notifications

To improve the user experience, it is suggested to implement the push notification option. The notifications alert the user if new episodes of the subscribed podcast have been uploaded. This option makes the manipulations with the application more convenient since the user does not have to check for new episodes manually.

KaiUI repository

The forked KaiUI repository was modified to fit the needs of this project. Some of these changes could be useful for other KaiOS developers. As a result of that, the author's KaiUI repository will be public. Once a pull request created to the original repository, it would be merged and made public.

Public repository for the application

In addition, to the public KaiUI repository, the application's repository will also be published. This will give other developers the opportunity to use parts of the application they might need.

Submitting to KaiStore

In the future, it is planned to submit the application to the KaiStore. In order to do that, the submission requirements [18] have to be fulfilled. These requirements were out of the scope of the present thesis.

7. Conclusion

The goal of this thesis was to create an application for KaiOS that allowed the user to listen to podcasts. The idea originated from the problem that there was no comfortable way to listen to podcast on the KaiOS devices. The main functionalities of the application were the ability to listen and subscribe to podcasts.

To develop the application, the waterfall methodology was used. The development had the following phases:

- requirements specification;
- application design;
- implementation;
- testing the application.

The requirements to the application were set by the author. The design was conducted following the KaiOS official guidelines. The application was implemented using iTunes Search API and indexedDB. During the testing phase, the overall usability of the application was examined.

The final version of the application has the main functionalities of listening and subscribing to podcast. In addition, the application meets all the requirements and limitations set at the beginning of the thesis. The application can be further developed adding the option to download episodes and push notifications. c The author plans to implement these features and then submit the application to the KaiStore.

8. References

1. Edison Research. Infinite Dial 2019 [Internet]. Data & Analytics presented at; 19:44:35 UTC [cited 2020 May 4]. Available from: https://www.slideshare.net/webby2001/infinite-dial-2019?ref=https://www.edisonresearch.com/infinite-dial-2019/

2. Rosin J. Puust ja punaseks: mis on podcastid ja kuidas neid kuulata? [Internet]. Digigeenius. 2018 [cited 2020 May 4]. Available from: https://digi.geenius.ee/rubriik/hea-nipp/puust-ja-punaseks-mis-on-podcastid-ja-kuidas-neid-kuulata/

3. Mobile operating system share worldwide from March 2019 to March 2020 [Internet]. StatCounter Global Stats. [cited 2020 Apr 26]. Available from: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201903-202003-bar

4. Introdcution | KaiOS Developer [Internet]. Developer Portal. 2020 [cited 2020 Apr 24]. Available from: https://developer.kaiostech.com/

5. JioPhone Operating System (Android or KaiOS or Firefox OS) [Internet]. 2017 [cited 2020 Apr 25]. Available from: https://web.archive.org/web/20171114084653/http://www.jio4gmobile.in/jiophone-operating-system/

6. Can I access your source code? [Internet]. KaiOS Support. 2019 [cited 2020 Apr 25]. Available from: https://support.kaiostech.com/en/support/solutions/articles/35000078432-can-i-access-the-source-code-

7. Elatta Leung. The Return of The Nokia 8110 – ‘the Banana Phone’ [Internet]. 2018 [cited 2020 Apr 26]. Available from: https://www.kaiostech.com/return-nokia-8110-banana-phone/

8. Explore KaiOS • Devices [Internet]. KaiOS. 2020 [cited 2020 Apr 25]. Available from: https://www.kaiostech.com/explore/devices/

9. Company • Our Story [Internet]. KaiOS. [cited 2020 Apr 27]. Available from: https://www.kaiostech.com/company/our-story/

10. Good news! India’s per-capita income rises 6.8 per cent to Rs 11,254 a month in FY20 [Internet]. The Financial Express. 2020 [cited 2020 May 8]. Available from: https://www.financialexpress.com/economy/good-news-indias-per-capita-income-rises-6-8-per-cent-to-rs-11254-a-month-in-fy20/1816070/

11. Report for Selected Countries and Subjects [Internet]. [cited 2020 May 8]. Available from: https://www.imf.org/external/pubs/ft/weo/2019/02/weodata/weorept.aspx?pr.x=38&pr.y=13&sy=2019&ey=2019&scsm=1&ssd=1&sort=country&ds=.&br=1&c=512%2C668%2C914%2C672%2C612%2C946%2C614%2C137%2C311%2C546%2C213%2C674%2C911%2C676%2C314%2C548%2C193%2C556%2C122%2C678%2

C912%2C181%2C313%2C867%2C419%2C682%2C513%2C684%2C316%2C273%2C913%2C868%2C124%2C921%2C339%2C948%2C638%2C943%2C514%2C686%2C218%2C688%2C963%2C518%2C616%2C728%2C223%2C836%2C516%2C558%2C918%2C138%2C748%2C196%2C618%2C278%2C624%2C692%2C522%2C694%2C622%2C962%2C156%2C142%2C626%2C449%2C628%2C564%2C228%2C565%2C924%2C283%2C233%2C853%2C632%2C288%2C636%2C293%2C634%2C566%2C238%2C964%2C662%2C182%2C960%2C359%2C423%2C453%2C935%2C968%2C128%2C922%2C611%2C714%2C321%2C862%2C243%2C135%2C248%2C716%2C469%2C456%2C253%2C722%2C642%2C942%2C643%2C718%2C939%2C724%2C734%2C576%2C644%2C936%2C819%2C961%2C172%2C813%2C132%2C726%2C646%2C199%2C648%2C733%2C915%2C184%2C134%2C524%2C652%2C361%2C174%2C362%2C328%2C364%2C258%2C732%2C656%2C366%2C654%2C144%2C336%2C146%2C263%2C463%2C268%2C528%2C532%2C923%2C944%2C738%2C176%2C578%2C534%2C537%2C536%2C742%2C429%2C866%2C433%2C369%2C178%2C744%2C436%2C186%2C136%2C925%2C343%2C869%2C158%2C746%2C439%2C926%2C916%2C466%2C664%2C112%2C826%2C111%2C542%2C298%2C967%2C927%2C443%2C846%2C917%2C299%2C544%2C582%2C941%2C474%2C446%2C754%2C666%2C698&s=PPPGDP&grp=0&a=

12. Mobile operating system share India from March 2018 to March 2019 [Internet]. StatCounter Global Stats. [cited 2020 May 3]. Available from: https://gs.statcounter.com/os-market-share/mobile/india/#monthly-201803-201903-bar

13. Mobile operating system share India from March 2019 to March 2020 [Internet]. StatCounter Global Stats. [cited 2020 Apr 26]. Available from: https://gs.statcounter.com/os-market-share/mobile/india/#monthly-201903-202003-bar

14. Architecture | KaiOS Developer [Internet]. Architecture. 2020 [cited 2020 Apr 26]. Available from: https://developer.kaiostech.com/introduction/architecture

15. Technologies | KaiOS Developer [Internet]. Technologies. 2020 [cited 2020 Apr 26]. Available from: https://developer.kaiostech.com/introduction/technologies

16 Marsic I. Software Engineering [Internet]. New Brunswick, New Jersey: Rutgers University; 2012 [cited 2020 Apr 26]. 613 p. Available from: https://www.ece.rutgers.edu/~marsic/books/SE/book-SE_marsic.pdf

17. Alshamrani A, Bahattab A. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. International Journal of Computer Science Issues. 2015 Jan;12(1):106–11.

18. KaiStore | KaiOS Developer [Internet]. KaiStore. 2020 [cited 2020 Apr 27]. Available from: https://developer.kaiostech.com/submit-to-kaistore

19. Guru Network Limited. Castbox [Internet]. 2020 [cited 2020 Apr 27]. Available from: https://castbox.fm/

20. OS ENV Setup | KaiOS Developer [Internet]. [cited 2020 May 4]. Available from: https://developer.kaiostech.com/getting-started/env-setup/os-env-setup

21. Petuhhov I. Tarkvaraprotsess, üldised protsessimudelid. [Internet]. 2008 [cited 2020

May 3]. Available from: http://www.cs.tlu.ee/~inga/SE_materjal/Tarkvara_protsess_2008.pdf?fbclid=IwAR27W1TrlXfLwJk8nxVINYKhWrjlRvXad4laOJesY16LwZRMKCtW3-d0iCM

22. UI Component | KaiOS Developer [Internet]. [cited 2020 Apr 30]. Available from: https://developer.kaiostech.com/design-guide/ui-component

23. Design Guide | KaiOS Developer [Internet]. Design Guide. 2020 [cited 2020 Apr 27]. Available from: https://developer.kaiostech.com/design-guide

24. Basic Navigation | KaiOS Developer [Internet]. [cited 2020 Apr 30]. Available from: https://developer.kaiostech.com/design-guide/basic-navigation

25. Development Environment | KaiOS Developer [Internet]. Development Environment. 2020 [cited 2020 Apr 27]. Available from: https://developer.kaiostech.com/getting-started/env-setup/development-env

26. WebStorm: The Smartest JavaScript IDE by JetBrains [Internet]. 2020 [cited 2020 Apr 27]. Available from: https://www.jetbrains.com/webstorm/

27. Chacon S. git-scm [Internet]. git. 2020 [cited 2020 Apr 28]. Available from: https://git-scm.com/

28. Lardiois F, Lunden I. Microsoft has acquired GitHub for $7.5B in stock [Internet]. TechCrunch. 2018 [cited 2020 Apr 29]. Available from: https://social.techcrunch.com/2018/06/04/microsoft-has-acquired-github-for-7-5b-in-microsoft-stock/

29. Simulator | KaiOS Developer [Internet]. Simulator. 2020 [cited 2020 Apr 29]. Available from: https://developer.kaiostech.com/getting-started/env-setup/simulator

30. FAQ — WHATWG [Internet]. [cited 2020 Apr 29]. Available from: https://whatwg.org/faq#what-is-the-whatwg

31. Cascading Style Sheets [Internet]. [cited 2020 Apr 29]. Available from: https://www.w3.org/Style/CSS/Overview.en.html

32. CSS Introduction [Internet]. [cited 2020 Apr 29]. Available from: https://www.w3schools.com/css/css_intro.asp

33. Nadim JR. nadim1992/KaiUI [Internet]. 2020 [cited 2020 Apr 29]. Available from: https://github.com/nadim1992/KaiUI

34. JS Foundation. jQuery [Internet]. [cited 2020 Apr 29]. Available from: https://jquery.com/

35. Reisig J. History of jQuery [Internet]. Technology presented at; 2007 Oct 31 [cited 2020 Apr 29]. Available from: https://www.slideshare.net/jeresig/history-of-jquery

36. jQuery Introduction [Internet]. [cited 2020 Apr 29]. Available from:

https://www.w3schools.com/jquery/jquery_intro.asp

37. IndexedDB API [Internet]. MDN Web Docs. [cited 2020 May 6]. Available from: https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API

38. iTunes Store API [Internet]. [cited 2020 Apr 30]. Available from: https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/

Appendix

I. Pictures of the Application Prototype

Figure 7: Application subscription view

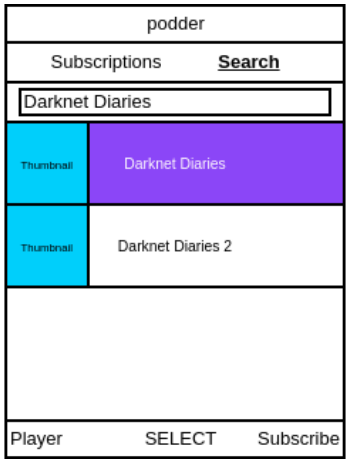

Figure 8: Application search view

Figure 9: Application podcast view

Figure 10: Application queue view

Figure 11: Application player view

II. Application Installation Guide

- Install a suitable version of WebStorm from jetBrains[2] website.
- Open the WebStorm program.
- Unzip the project to your preferred directory.
- Load the unzipped file folder to WebStorm.
- When prompted with the IDE, right click on the "index.html" file, and click run.
- It opens up in your default browser.

[2] https://www.jetbrains.com/webstorm/download/

III. Additional Files

KaiOS_podcast.zip

IV. License

Non-exclusive license to reproduce thesis and make thesis public

I, **Fredi Arro**, herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

KaiOS Application for Listening Podcasts

supervised by Ljubov Jaanuska.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Fredi Arro

Tartu, **08.05.2020**