UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Joosep Tenn

Errand Running App for Android With Geolocation Support

Bachelor's Thesis (9 ECTS)

Supervisor: Eero Vainikko, PhD
Supervisor: Amnir Hadachi, PhD

Tartu 2017

Geoasukoha toega tööotsade tegemise rakendus Androidile

Lühikokkuvõte: Bakalaureusetöö käigus loodi rakendus nimega ErrandMe, mis võimaldab inimestel pakkuda ning otsida ajutisi tööotsasid. Tööotsad on lühiajalised tegevused, mille tegemise eest on tihtipeale ette nähtud rahaline tasu. Rakenduse abiga on võimalik kokku viia inimesed, kes otsivad ajutist tööd kui ka need, kes vajavad abikäsi ning on nõus abistajale vaevatasu maksma. Kasutajate pakutavatel tööotsadel on pealkiri, kirjeldus, tasu koos valuutaga, toimumise asukoht ning tööpakkuja soovil ka hinnanguline kestvus. Oluline osa rakendusest on geoasukoha kindlakstegemine. Kui kasutaja lubab rakendusel kasutada asukoha tuvastamise funktsionaalsust, siis on võimalik lisada ja otsida tööotsasid nutitelefoni asukoha põhjal, mis näitab kauguse tööotsa toimumiskohast.

Võtmesõnad: Android, rakendus, ülesanne, raha, tasu, asukoht, nutitelefon, Firebase

CERCS: P170 Arvutiteadus, arvanalüüs, süsteemid, kontroll

Errand Running App for Android With Geolocation Support

Abstract: In this thesis, errands are considered as tasks that are paid for upon completion. Some people are willing to run an errand if they are paid, and some people need help with an errand and are willing to pay the person or people who complete it. With this thesis, an Android application called ErrandMe was developed to satisfy the needs of both groups. Every errand has a title, description, pay, currency, location and optionally an estimated completion time. Geolocation is an important part of ErrandMe. If allowed, the application tracks the location of the user’s smartphone, which can be used to add and find errands, but also display the distance from the location of an errand.

Keywords: Android, application, errand, money, pay, location, smartphone, Firebase

CERCS: P170 Computer science, numerical analysis, systems, control

Table of Contents

1 Introduction

A great number of activities that many people have to do can be considered as errands. Taking out trash, changing a baby's diaper, but also buying food for a pet can be an errand. These activities might not be enjoyable, but often they just have to be done. A baby does not know how to change his or her diaper, nor does a dog go shopping for food. Errands can be short, for example, changing a broken light bulb, but they can also be quite time consuming, for example, transporting a table to a location that is a few hundred kilometers away. Sometimes, especially when long errands are concerned, a person might want to have someone help run, or better yet, let someone do the whole errand without any help. The idea of letting someone else run errands is reasonable. Some time could be saved, and it would be possible to concentrate on more important activities. It is naive to expect to find someone who is not a relative or a friend and who is willing to run them without any type of reward. If time and effort are val- ued, some money could be spared to hire a stranger to run the necessary errands.

1.1 Problem statement and motivation

Humans have a necessity to optimise their sources of income in order to sustain their financial needs. As time goes by, life becomes more expensive, and higher incomes are required to maintain the same quality of life. The truth of the matter is, everything is becoming more expensive, but incomes are usually not catching up. To avoid making sacrifices in the quality of life, a person has to find new ways to improve their financial situation. Most of the time, the solution is to find more or higher paying work. Finding work is not a simple process as it often requires specific knowledge and experience. Higher paid jobs usually require diplomas and higher education, which many people do not have. On top of that, there are usually a great number of people who are qualified to perform certain tasks, but not all of them are accepted by companies. Because of these factors, there are some people who have a hard time finding work and earning money.

There are also people who have the opposite problem, finding people to do some work. Anyone can relate to lifting or moving something that is too heavy or having to run several errands at the same time. These are the moments where it would be excellent to have a helper. Should it be a grandmother needing help with storing firewood into a shed or a music student moving into her new rental space together with her grand piano.

A solution that would deal with both of the mentioned issues would be to create a mobile application that allows providing and searching for some short-term work. Even though similar applications have already been made, such as TaskRabbit, there are possibilities for making these applications more user-friendly and better in general. While TaskRabbit has issues with user equality and inflated worker prices, some other solutions have a specific field of use that restricts the work that could be done.

An application could be created that provides equal conditions for all of the users and allows advertising and searching for errands of any kind with the possibility to use the user's location. A solution that makes it possible to find work at any distance, pay and completion time and provide work to anyone who is interested or qualified.

1.2 Objectives and limitations

The aim of this thesis is to create an android application, ErrandMe, for creating, searching and accepting errands while providing geolocation support. The concept of ErrandMe is straightforward; someone uploads an errand and waits for someone else to accept the uploaded errand. Both the person who accepted the errand and the person who created the errand will receive contact information of the other person after the errand has been accepted. The contact information can be used to schedule a time for doing the errand. Geolocation and location awareness allow a user to find nearby errands, upload new ones based on the device's current location and track the distance from accepted errands.
ErrandMe is meant to provide a way to earn money and help people with their daily activities. People who could benefit the most from the errand creating functionality are the elderly and people with disabilities. People who could benefit the most from accepting errands might be college students and young people without a job.

ErrandMe also has some limitations. Some are because of the time and size constraint of the thesis, and some are caused by architectural decisions. In Estonia and some other countries, there exist laws that prohibit providing work to persons who have not been registered in an employment register or verified in some other way. There is no functionality in the application that allows worker registration or checking a person's background. Although, in Estonia there is also a simple way to register workers that requires making a phone call or sending a text message [1]. This approach could be used in ErrandMe, but it is not easy to force this behaviour. A better system should be implemented.
ErrandMe is made for smartphones with Android version 4.0.3 or higher, which covers around 99% of the devices running an Android platform [2].
A Backend as a Service, Firebase, acts as a database and a server. For ErrandMe, Firebase's free plan is used, which limits the amount of simultaneous connection and the amount of data that can be stored [3]. Premium plans allow more data storage and an unlimited amount of simultaneous connections [3].

1.3 Outline

Chapter 2 covers the state-of-art of the technology used. Applications that have already been made in the field are analysed.
Chapter 3 provides an in-depth analysis of ErrandMe. Functional and nonfunctional requirements are covered. The design and architecture of the application are described, and the methodology used in development is explained.
Chapter 4 offers an overview and analysis of the results. Application performance is measured. Questions are asked from the users and tests are performed to evaluate the validity of the non-functional requirements and general application quality.
Chapter 5 provides a conclusion to the thesis. Future work and improvements that can be made are covered.

2 State-of-the-art

2.1 Introduction

Mobile application development is a field with great potential. It is mainly because of the fast growth and development of mobile devices, which make these devices more and more powerful over time. Not only does the software in these devices become more powerful, but also efficient. As new functionalities are added to the smartphones, mobile developers can use it to make new applications or improve existing ones. For example, sensors in smartphones have made it possible to track footsteps, identify fingerprints and measure temperature. On top of that, there are several mobile platforms, integrated development environments and other technologies that simplify application development.

In this section the technologies used for developing the errand running application are described and evaluated. Similar applications that are currently in use are reviewed.

2.2 Technologies

2.2.1 Smartphones

Smartphones are mobile phones that have significantly more functionality than a regular mobile phone. They are mobile computers. What makes them special is flexibility of use; they can be used for more than just calling or messaging someone. Smartphones can be used for entertainment, for example, watching videos, movies or reading online newspapers. They can be used to hold business meetings between many people via Skype or some other application. They can be used as a measuring tool for measuring foot steps, air temperature or pressure, and much more. Smartphones are powerful and flexible because they have built in sensors, powerful processors, multiple network interfaces and a high amount of memory for such small devices. For example, Samsung Galaxy S8 has a 2.3 or 2.35 GHz quad-core processor, up to 256 GB of external memory and 4 GB of RAM [4]. It is one of the best smartphones currently available [5]. An average personal computer has similar technical specifications.

The main alternatives to smartphones are computers. Smartphones were chosen as the target devices for ErrandMe because of their flexibility of use. Mobile phones can be carried anywhere and do not require much space, whereas computers are bigger and harder to transport.

2.2.2 Android

Android is a Linux kernel based open-source mobile operating system developed by Google for phones, tablets, watches, TVs, cars and other electronic devices [6]. Being open-source, everyone have full access to the Android source code, with one restriction, it cannot be used for personal profit or any financial gain. It is the most popular mobile operating system [7].
Like with any other mobile operating system, an important part of Android is the ability to install and run applications. The primary source being the Google Play Store where there are millions of applications available [6]. But there are also other marketplaces where Android applications can be installed from, such as Samsung's Galaxy Apps and Amazon Appstore. Some applications have a free and a paid version, and some applications are either free or only have a paid version. Paid versions usually come with more functionality and provide a

better user experience than their free counterparts. Free versions of applications that have paid variants usually only provide the core functionalities.
Since the beginning of Android, many versions have been released, the newest being Nougat (version 7.0 - 7.1) [2]. Usually newer versions provide developers with new libraries and functionality that they can use when creating applications.

Android alternatives include iOS by Apple, Windows Phone, BlackBerry, Symbian and a few others. The author chose Android as it is the operating system that the author has the most programming experience with. Android’s market dominance and cheap application release costs were also deciding factors.

2.2.3 Android Studio

Android Studio is the primary Android IDE (Integrated Development Environment). It provides an Android developer all the necessary tools to develop an Android application. More specifically, it allows writing code with auto-completion tools, debugging, testing, running the code on a physical or a virtual device and setting programming related or visual preferences. The virtual devices are provided by the Android Emulator. The emulator makes it possible to run an application on a large variety of Android devices with different configurations. A developer can choose between different device manufacturers, newer and older Android versions and even different devices such as tablets, mobile phones, watches and TVs. Android Studio makes user interface design simple by providing a GUI (Graphical User Interface) where it is possible to drag all the necessary elements into the XML layout instead of writing XML by hand. Furthermore, it provides tools to analyse what is inside the Android Package Kit and create image assets with different densities or use existing ones provided by Android Studio. Java and XML are the only languages required to create Android applications with Android Studio. [8]

Android Studio does not have any alternatives worth considering. It is possible to develop Android applications with Eclipse by using the Android Developer Tools plugin, but it is no longer supported by Google [9].

2.2.4 Java

Java is a “general-purpose, concurrent, class based, object-oriented” programming language [10:1]. It is a high-level, strongly typed language with garbage collection that incorporates concepts from several languages including C and C++, but it is not entirely the same. For example, Java does not allow writing unsafe code that might cause vulnerabilities and unexpected behaviour. The main building blocks of a Java application are classes, interfaces and packages. Unlike some other object-oriented languages, Java does not allow multiple inheritance; every class can only have one parent class. Although, multiple inheritance is possible with interfaces. Programs written in Java are compiled to bytecode and can be run by the Java Virtual Machine, a virtual computer that makes it possible to run Java programs. [10]

Java, Lua, JavaScript and a few other programming languages can be used to develop android applications. The author chose Java as the primary programming language because it is used in Android Studio.

2.2.5 XML

Extensible Markup Language (XML) is an alternate form of storing and sharing structured data online compared to HTML (Hypertext Markup Language). It is a markup language that is rather similar to the Standard Generalized Markup Language, but does not have all the same features. It has a more simple implementation, and it is meant to be easier to understand and use. XML documents are both human and machine readable, and they contain components called elements. All the elements begin and end with tags and can contain attributes, which are pairs of names and values. XML documents have a tree-like structure. They contain one root element, which can contain sub-elements. All sub-elements can also contain sub-elements. [11]

The author used XML in the development process due to it being the markup language used in Android Studio to create layouts.

2.2.6 Gradle

Gradle is a build system that is based on the JVM (Java Virtual Machine). Gradle relies on a domain-specific language that is Groovy based, whereas other build tools usually use XML. Gradle has its own and flexibly configurable dependency management that enables the use of different libraries and frameworks in the code. [12]

Alternative build systems for Java-based applications include Apache Maven, Apache Ant, Apache Buildr and some others. The author chose to use Gradle as the author has the most experience with it out of all the build tools, and it is also a part of Android Studio by default.

2.2.7 Global Positioning System

Global Positioning System (GPS) is a navigational system of at least 24 satellites that orbit the Earth. The satellites are equally divided between 6 orbital planes. An object located on the Earth with a GPS receiver can receive accurate position and velocity data from the system, regardless of the time. The GPS receiver that has been built into an object is responsible for finding the object’s position. To determine an object’s position, TOA (Time of Arrival) ranging is used. GPS is used widely in many different areas. It was initially meant for military use, but today it is used as a travelling guide, a tool for organizations and people to track other people or objects, and much more. [13]

Besides GPS, cellular towers and network connection can be used to detect a user’s location, both of which are also used in ErrandMe, depending from a user’s location and network related settings. GPS usually provides the highest accuracy, but sometimes it is not possible to detect a device’s location, especially indoors. GPS usage is also more battery consuming compared to the alternatives, but as ErrandMe prioritises accuracy in location detection, it is used when possible.

2.2.8 Google Play services

Google Play services provide application developers a comprehensive set of useful features, for example, Maps and Google+ sign-in. The services include the Google Play services client library and the Google Play services Android Package Kit. The client library makes it possible to access any feature with a user’s account and deals with different issues that may occur when using the services. The Android Package Kit communicates with the client library and provides access to a specific service when necessary. Google Play services can be installed from the Google Play Store. [14]

The Google Play services contain most of the core services for Android, and there are no real alternatives. The use of Google Play services is a must when using Firebase. Important functionalities in ErrandMe, for example, viewing an errand on a map and obtaining a user's location also rely on the services.

2.2.9 Backend as a Service

Backend as a Service (BaaS) is a hosted backend that has been premade for developing a web or mobile application. Developers do not have to write any or much backend specific code. It has all the necessary features of a backend and even more. Features such as Facebook and Google sign-in integration and cloud messaging are common. The features can be accessed by documented APIs that simplify the application development process. Currently there are many different BaaS providers to choose from, and usually it is possible to choose between different plans. Providers usually have one free plan and one or more premium plans. Premium plans provide more storage and do not have as strict limits as free plans.

The only real alternative to using a BaaS is using a real server. A BaaS based approach was chosen by the author as it requires less work. Setting up and managing a server is time consuming and can be quite troublesome, for example, handling scalability and making sure that the data is always available for the users.
BaaS also has a few issues. Usually there are quite large restrictions when free plans are used. The restrictions usually include less available storage space and a smaller number of simultaneous connections to the database. Although there are restrictions, free plans are sufficient when an application does not have many users.

2.2.10 Firebase

Firebase is one of many implementations of the BaaS model. Like other BaaS implementations, Firebase provides storage, push notifications, user authentication and a database. Other than the basic BaaS features, Firebase also provides a test lab that allows testing a Firebase linked application with various configurations and devices. A feature that makes Firebase different from other BaaS implementations is the real-time database. When new data is added to the database, it becomes accessible instantly to all the users of the application. Firebase can be used for Android, iOS, Web, C++ and Unity development. [15]

There are many BaaS platforms to choose from, for example, Kumulos [16] and Kinvey [17]. Not all of the BaaS platforms are meant for creating real-time applications, but Firebase is, which is one of the reasons why Firebase was chosen by the author. Although Firebase is not the only BaaS that has a real-time database, Firebase is well documented and has a satisfactory set of features in the free plan, which also contributed to the choice.

2.3 Available solutions

ErrandMe is not an original idea. Several mobile and web applications that deal with errand running, work providing and doing chores have been created. These applications seem to be quite popular in the United States as several alternatives exist. In Europe, on the other hand, the competition is small as only a few solutions are available.

2.3.1 GoWorkaBit

GoWorkaBit is a work providing application that has become successful in Estonia and is widely used. With GoWorkaBit, it is possible to filter work by distance, pay and date. However, as Estonian law requires the registration of workers, it is necessary to provide some personal information before it is possible to find any work. GoWorkaBit requires initial users to provide their personal information, a photo and also prove that they are allowed to work in the country. In Estonia, GoWorkaBit will automatically notify the Estonian Tax and Customs Board about a person finding work. Only organizations are allowed to provide work, which should help deal with worker absences within the organization. The work seems to be oriented toward young people without enough work related experience, but there is also work that requires more skill and knowledge. There is work that can be done in less than a day, but there is also work that takes more than a week. GoWorkaBit tries to suggest work to users instead of making them search for available work themselves. That is done by sending notifications and emails to the user. Currently, GoWorkaBit is only usable in Estonia and Lithuania, but is planning to expand to other European countries in the near future. GoWorkaBit is currently not available as a mobile application. [18]

2.3.2 TaskRabbit

TaskRabbit is a rather successful errand running application that started in the United States. Before a person can become a tasker, a background check is performed, and the candidate is either met in person or interviewed. The worker can be rated and reviewed after completing each task. The average rating and reviews can be used to build a reputation as people will look at them when hiring a person. Workers with the highest rating and with many tasks completed are usually highlighted, which improves their chances of finding work.

When looking for a worker, a person has to first enter the street address, which will be checked for availability. Then a brief description about the work has to be provided. After that, the person can either choose a worker or let the system notify workers with the provided description. When choosing the latter, an hourly price is provided and the employer cannot choose the pay. Choosing the other option means that a worker with a suitable hourly price and rating can be selected. The workers can be sorted by rating, price, reviews and recommendations. It is also possible to chat with clients and hire several people to complete a task. Workers have the freedom to choose their hourly rate and change it. TaskRabbit will take a 30% service fee from the total pay of every task. Workers can mark their desired work area on a map by drawing a shape on the map. It will be used to inform the user of available work and help employers choose a suitable tasker. TaskRabbit is currently available in the United States and the United Kingdom, but is also planning to expand in the future. It is available as a web, Android and iOS application. [19]

2.3.3 Agent Anything

Agent Anything is an errand running application that aims to provide work for students. There are no restrictions to the tasks, anything can be done. Tasks can be provided by both individuals and companies. Every task is visible on a map. Taskers have to verify their student status before they can accept any tasks. The pay of tasks can be preset or negotiable.

Agent Anything is currently available in New Jersey, New York and Adelaide as a web application. [20]

2.3.4 Other solutions

Most of the available work finding and providing applications are meant to help enterprises. The main goal is usually to help find people quickly to fill in shifts or deal with worker absences. Only a few solutions present individuals a chance to provide work, and even then the tasks are very specific and mostly housework related.

2.4 Conclusion

Many workforce providing applications have been created, but a large number of them have a specific field of use. Some are meant to help enterprises find short term workforce, and some are meant to help individuals with errands in the house. Only TaskRabbit seems to stand out from the rest as it is centered around individuals of all ages and background. Anyone can search for and provide tasks. The activities that have to be done can be anything. There are still a few problems with TaskRabbit. First of all, there is a problem with user equality. As people with a higher reputation and more experience have a higher chance of finding tasks, users with less experience have a lower chance of finding work. Secondly, by charging 30% for every task, the hourly rates of taskers are inflated and users looking for helpers might become hesitant because of the high prices. If it is not possible to cover all the costs with smaller service fees, a different approach should be used. Instead of letting work providers cover all the fees, people searching for work should pay something as well as long as it is still profitable for the users. This approach together with ad revenue should cover all the costs and balance the payments between different groups.
ErrandMe is an errand running application that tries to deal with the issues of TaskRabbit and use a different approach for searching and providing errands. It is an application that provides everyone an equal chance to run errands and will use ad revenue to cover all the fees as long as possible. A solution for Android smartphones that is centered around the use of the user's location and Firebase.

3 Mobile application design and architecture

3.1 Introduction

This chapter discusses the creation of ErrandMe. First of all, the functional and non-functional requirements of ErrandMe are discussed. Secondly, the design of ErrandMe is covered where different views are looked at. Navigation inside the application is explained. Thirdly, the architecture of ErrandMe is examined where the role of Firebase and communication between the database and the application is covered. Next, the methodology is explained where all of the classes of ErrandMe are introduced and methodology used to make the application work is analysed. Last but not least, a conclusion is provided.

3.2 System design and architecture

3.2.1 Functional requirements

The functional requirements of ErrandMe are listed in table 1. The listed functionalities describe the application's capabilities and the activities that the users of the application should be able to perform.

Table 1. Functional requirements for ErrandMe

ID	Requirement	Priority
1	The user can create a new account. Users should provide their email address, phone number, full name and a password when registering a new account.	High
2	The user must be able to log in with a registered account.	High
3	The user must be able to add new errands. Every errand should contain a title, description, pay, currency and location. Optionally an errand may contain the estimated errand completion duration.	High
4	The user should be able to edit active added errands.	High
5	The user must be able to search for available errands. Searching should be done based on a user entered search radius from a user entered search location. It must be possible to filter the errands by pay, estimated completion time and words or phrases in the errand title.	High
6	The user must be able to accept available errands.	High
7	The user should be able to use his or her location when adding new or searching for existing errands.	High

8	The application should not display any active accepted errands when searching for errands.	High
9	The user should be able to view his or her active errands. There should be separate views for added errands and accepted errands.	High
10	The user should be able to change his or her contact email and phone number.	Medium
11	The user should be able to choose the contact information that is shared with other users. It should be possible to share only the phone number, only the email address or both.	High
12	The user must be able to decline an active accepted errand.	High
13	The user must be able to remove an active added errand.	High
14	The user must be able to view an other user's shared contact information if the users share an errand (one as the owner and one as the accepter).	High
15	The user must be able to view the location of his or her active accepted errands on a map.	Medium
16	The user must receive a notification in the notification bar if his or her active added errand is accepted or declined. A notification should also be sent when the user's accepted errand is removed by the errand owner. Notifications should only be sent when the user is logged in. If the state of an errand changes during the time a user is not logged in, no notifications should be sent when the user logs in.	Medium
17	The user must be able to log out of the application.	High

The listed requirements concern only the main functionalities of the application. These requirements must be satisfied. ErrandMe has other functionalities that are essential for any application. For example, checking the availability of internet connection and validating user input.

3.2.2 Non-functional requirements

Non-functional requirements describe the limitations of the application, the expectations for the users and the performance of the application. The requirements are as follows:

1) The application must run on all Android smartphones with Android version 4.0.3 and above;
2) The user must have access to the internet to use the application;
3) The user must have the latest version of Google Play installed on his or her Android smartphone;

4) At least 4 users out of 5 should have no difficulty in using and navigating between any of the functionalities of the application without any external assistance;
5) At least 4 users out of 5 should be able to search and find a suitable errand within 2 minutes if it exists and has not been accepted yet;
6) Errand and user data changes should reach the user within 4 seconds after they have been made.

The validity of the listed non-functional requirements are analysed in chapter 4.

3.2.3 Application design

The Android application consists of the following views:

1) User log in
2) User registration
3) Active added errands
4) Active accepted errands
5) Errand adding
6) Errand searching
7) Search results
8) Settings

User login view is the view that is displayed if the user is using the application for the first time or has not logged in. It is possible to log in with Google or an account created in the registration view. If the user chooses the latter, email and password have to be entered. If the user has not yet created an account, there is a button that takes the user to the registration view. The layout of the login view is visible in figure 1, and the Google sign-in dialog is visible in figure 2.

User registration view allows the user to create a new account. The view is displayed after pressing the "Sign up!" text in the login view. When registering an account, the user has to enter his or her full name, phone number, email address and choose a password. The email address is used to log in to the application, but it can also be shared with other users as a communication tool. The entered email has to be unique. The password must be between 8 and 25 characters long. The layout of the registration view is visible in figure 1.

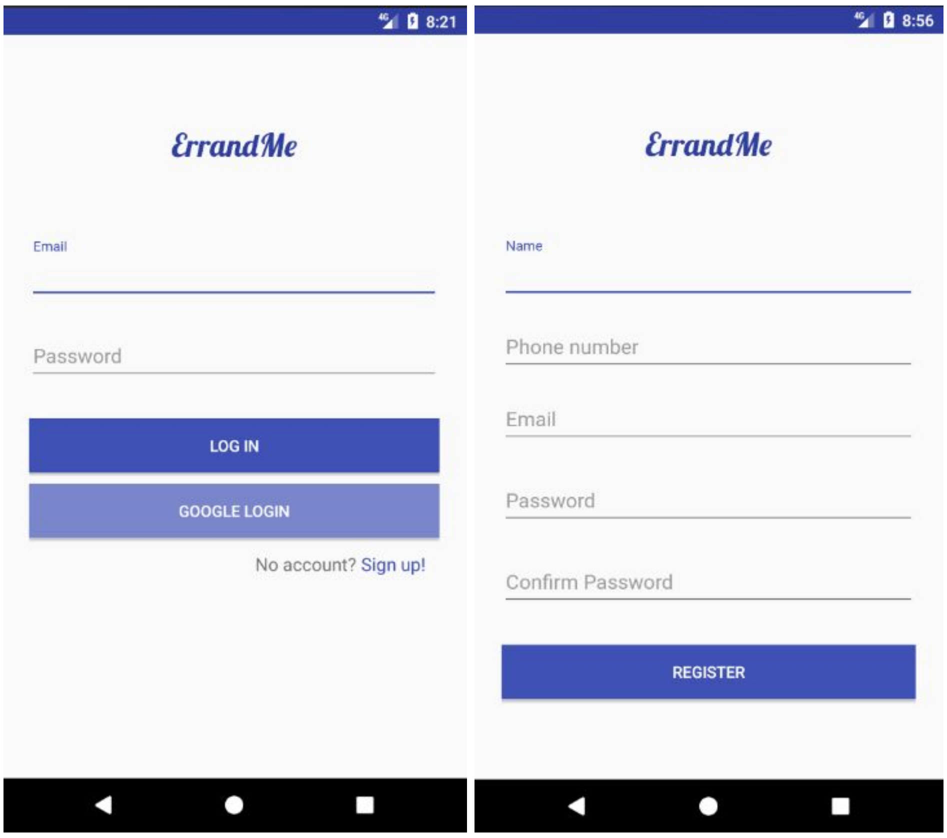


Figure 1. Login view on the left and registration view on the right.

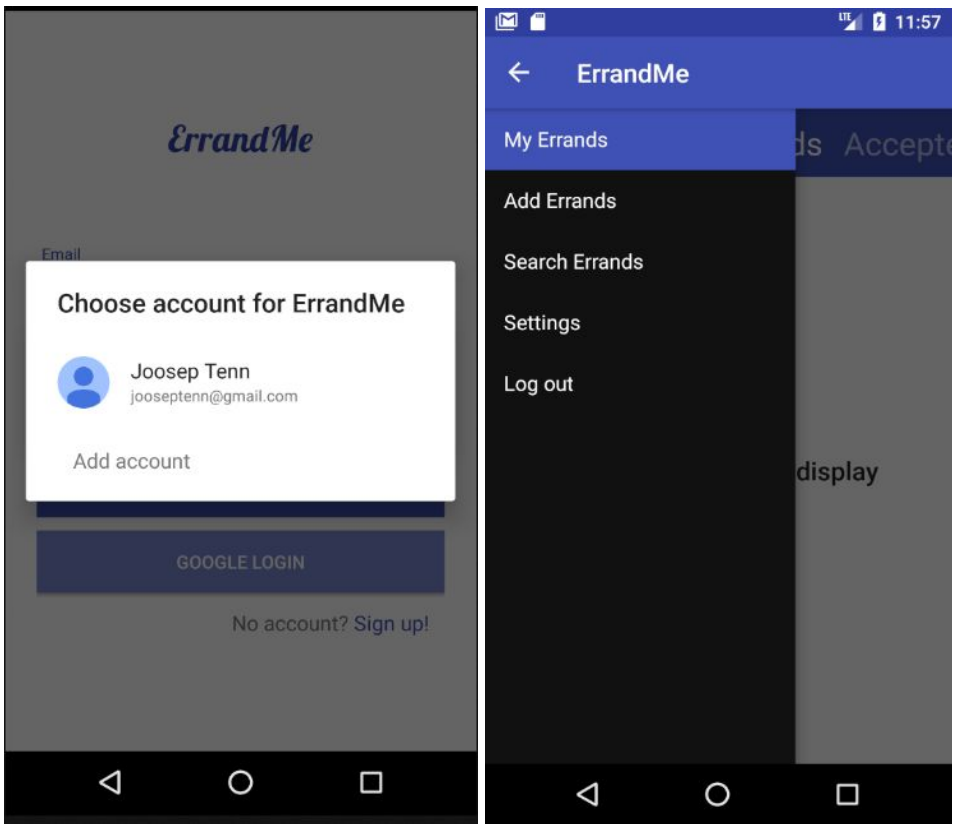


Figure 2. Google sign-in on the left and the navigation drawer on the right.

Active added errands view is displayed when logging in to the application, choosing "My Errands" from the navigation drawer (figure 2) or when a user has already logged in and brings the application to the foreground. It displays all of the active (not removed) errands that the user has added. Every errand has two possible states: accepted or still available. If an errand is still available then the user can view its details (figure 4), edit (figure 5) and remove the errand. When the errand is accepted by someone, the editing functionality becomes unavailable, and the accepter's contact information can be viewed (figure 4) by pressing the contact button. The layout of the active added errands view is visible in figure 3.

Active accepted errands view is displayed to the user after a swipe gesture to the right on the screen while being in the active added errands view. This view displays all of the active errands that are currently accepted by the user. The view has buttons for viewing the errand's details (figure 4), errand assigner's contact information (figure 4), the errand's location on a map (figure 5) and declining the errand. Every accepted errand also displays the distance from the user's current location, the pay given upon completion and estimated completion time.
If the user has enabled the use of GPS then the distance from the errand's location updates after every minute. The layout of the active accepted errands view is visible in figure 3.

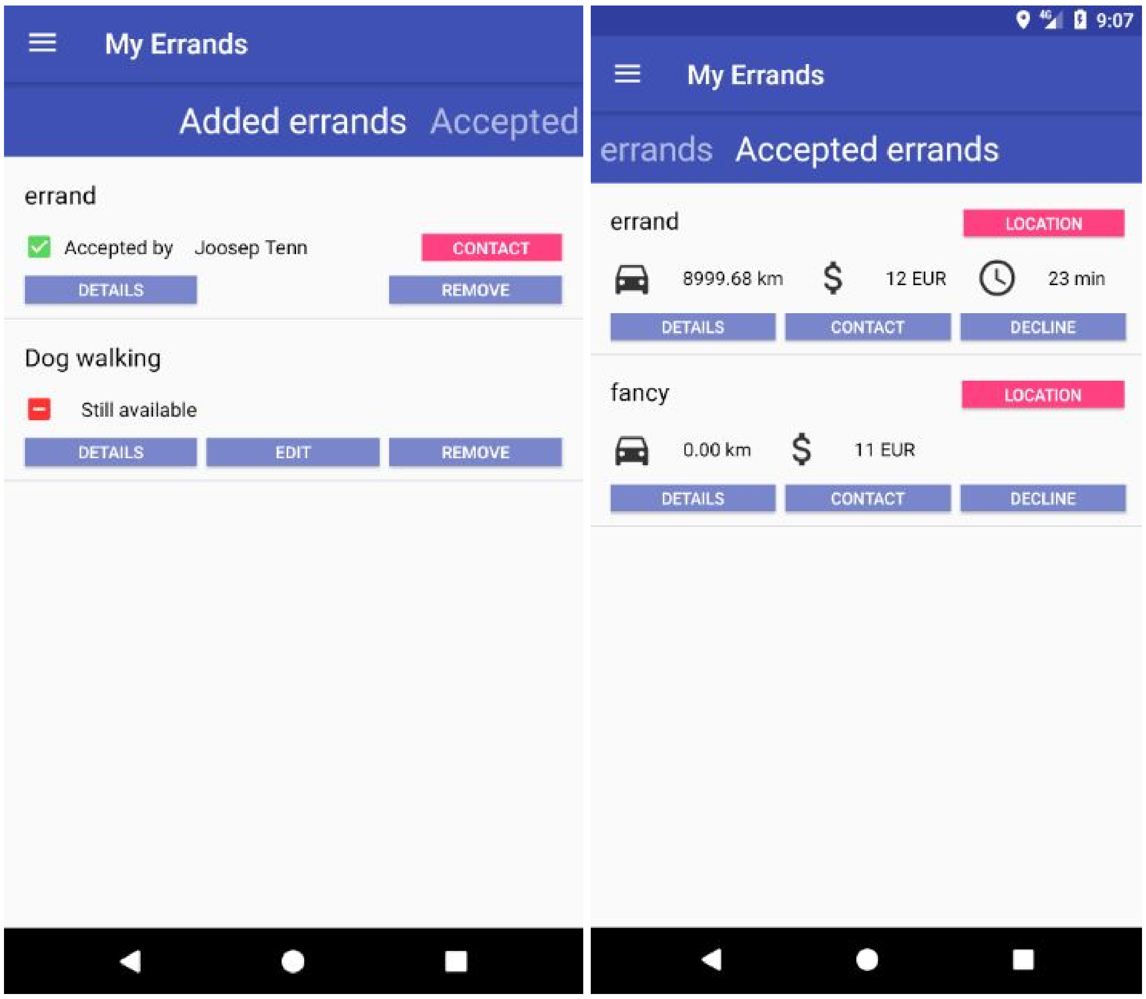
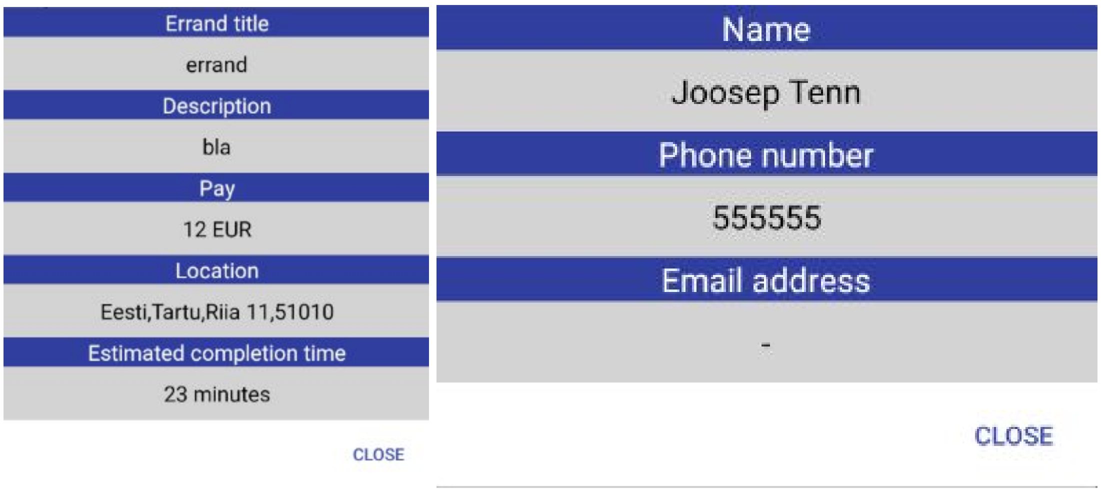


Figure 3. Active added errands view on the left and active accepted errands view on the right.


Errand title
errand
Description
bla
Pay
12 EUR
Location
Eesti,Tartu,Riia 11,51010
Estimated completion time
23 minutes
CLOSE

Name
Joosep Tenn
Phone number
555555
Email address
-
CLOSE

Figure 4. Errand details dialog on the left and contact information dialog on the right.

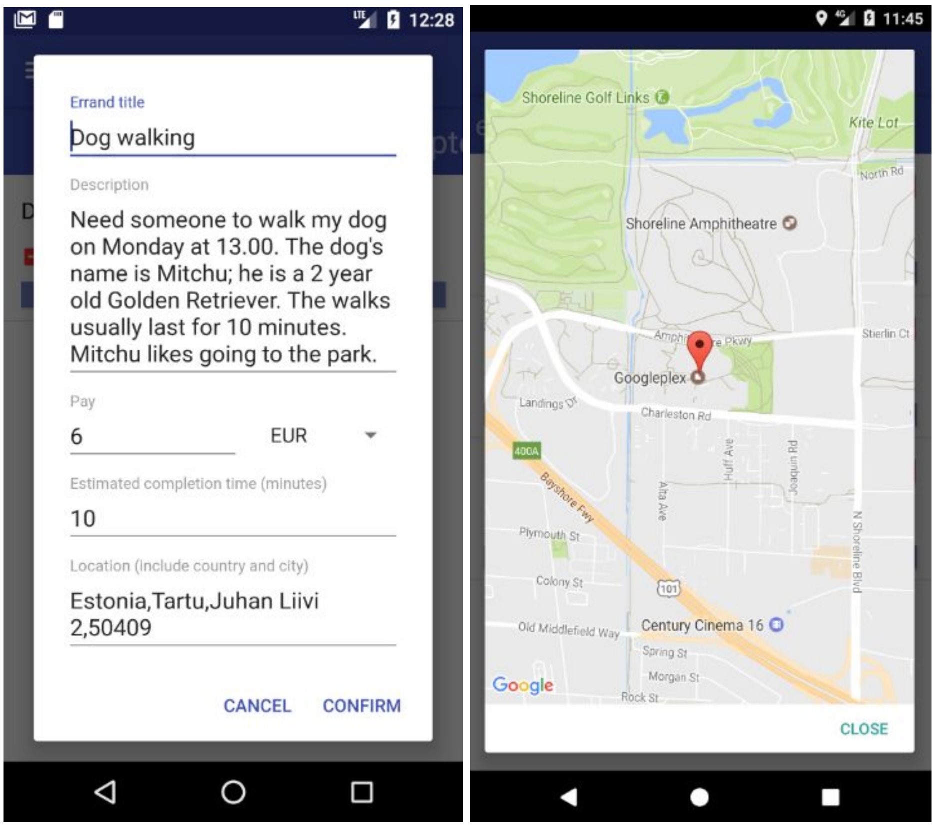

Errand title
Dog walking
Description
Need someone to walk my dog on Monday at 13.00. The dog's name is Mitchu; he is a 2 year old Golden Retriever. The walks usually last for 10 minutes. Mitchu likes going to the park.
Pay
6 EUR
Estimated completion time (minutes)
10
Location (include country and city)
Estonia,Tartu,Juhan Liivi 2,50409
CANCEL CONFIRM

Figure 5. Errand editing dialog on the left and errand location dialog on the right.

Errand adding view is meant for adding new errands. The user can navigate to the view by selecting “Add Errands” from the navigation drawer. To add a new errand, the user has to specify the title, description, pay, currency and location of the errand. If the user has allowed the application to track the device’s location then it is possible to use that location as the errand running location. The errand’s location will not change after the errand has been added, but it can be changed by editing the errand. The device’s location is used if the “Use current location” checkbox is checked. When entering the location manually, the user should specify the address of the location where the errand has to be done. The entered address should be as specific as possible. Depending on the errand type, the user may also decide to add an estimated completion time. By default, the “Use time estimation” checkbox is checked, which means that an estimated completion time has to be entered. The layout of the errand adding view is visible in figure 6.

Errand searching view allows users to search for active errands that have not been accepted yet. The user can navigate to the view by selecting “Search Errands” from the navigation drawer. When searching for errands, it is only mandatory to specify the search location and search radius in kilometers, which will be used to find the available errands in the area. The search radius has to be entered every time. If the “Use current location” checkbox is checked, the user can use his or her location to search for errands. If the checkbox is not checked or the user’s location cannot be determined, the location has to be entered manually. The user can filter the available errands by the words or phrases in the title, the minimum pay, currency and estimated completion time. If the “Enable search” checkbox is checked then it is possible to filter errand titles by the comma separated words or phrases entered in the first text field of the view. If the “Must match all” checkbox is also checked, the title of an errand has to contain all of the entered words and phrases to not be filtered out. If the checkbox is not checked, however, the title of an errand has to only contain one of the entered words or phrases. By enabling the “Enable pay filtering” checkbox, errands that have a smaller pay than entered in the “Minimum pay” text field are removed from the search results. If the “Enable time filtering” checkbox is checked, errands are removed from the search results based on their estimated completion time. Errands that have a longer estimated completion time than entered are not shown. Errands that have no estimated completion time are also not shown if time filtering is enabled. The layout of the errand searching view is visible in figure 6.

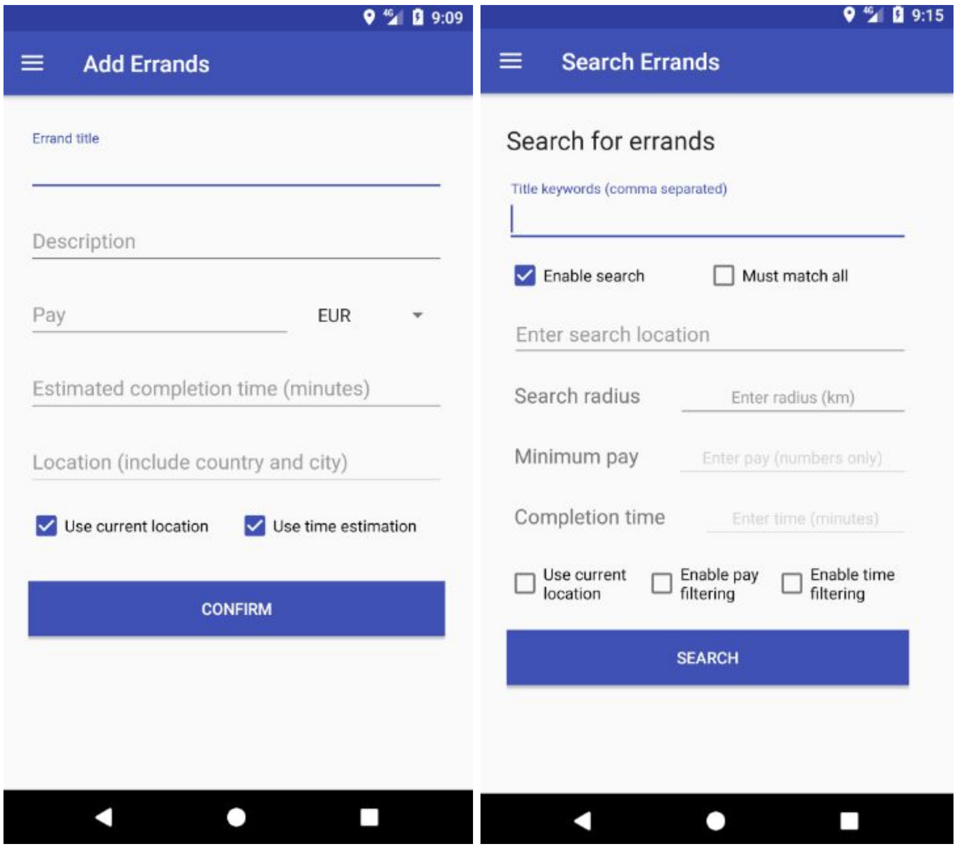
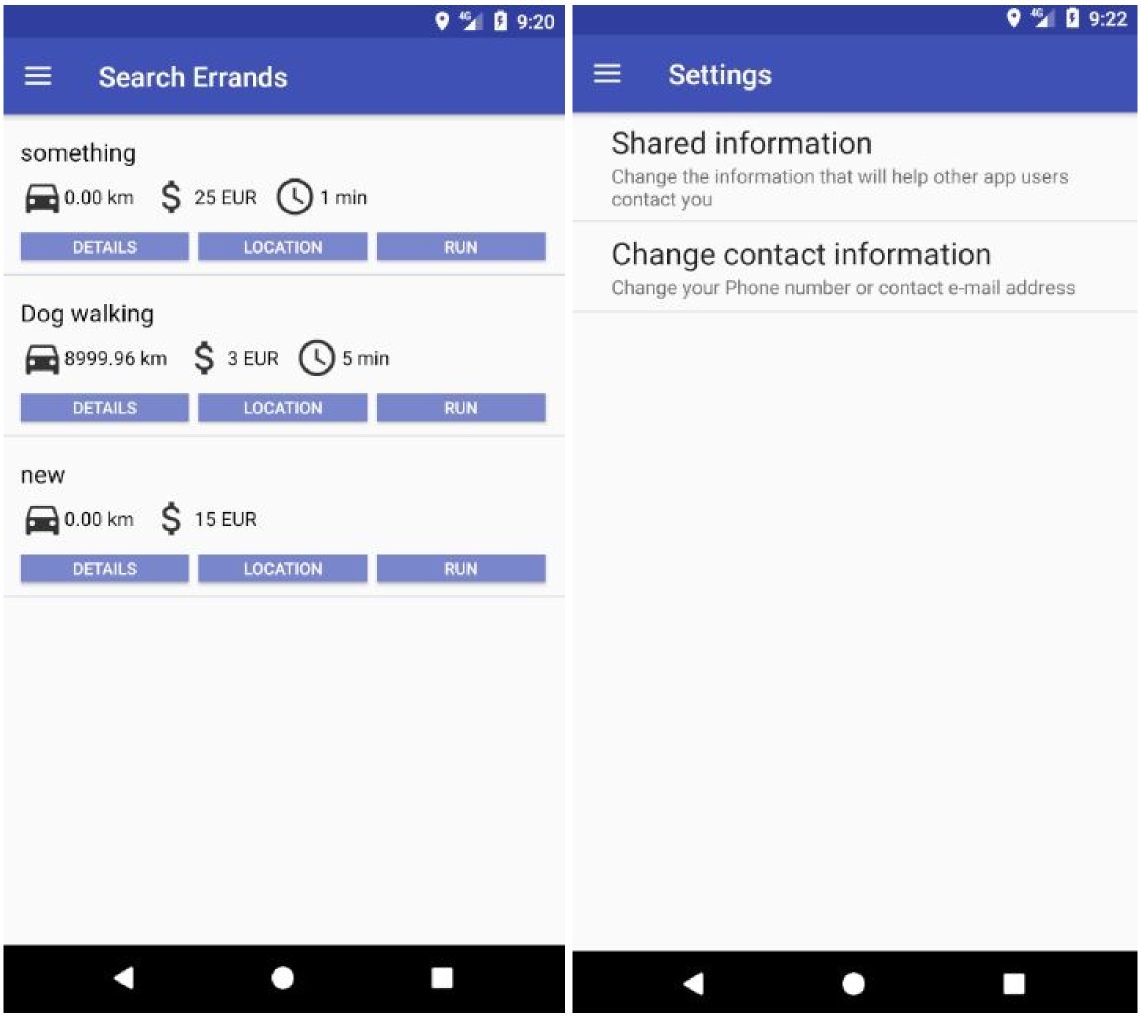


Figure 6. New errand adding view on the left and existing errand searching view on the right.

Search results view is displayed to the user after pressing the “SEARCH” button in the errand searching view. If any errands match the search criteria then the matched errands will be displayed one under another. The title, errand location, distance from the user, pay, currency and estimated completion time are displayed with every matching errand. The user can also view the errand’s details, check its location on a map and accept the errand. To accept the errand, the user has to press the “RUN” button. After accepting an errand, it will be automatically added to the user’s active accepted errands view.
If there are no matching errands, a text is displayed to the user that informs about no matching errands existing. The layout of the search results view is visible in figure 6.

Settings view is displayed if the user chooses “Settings” from the navigation drawer. The settings allow a user to change his or her phone number and email (figure 8) and choose the contact information that will be shared with other users (figure 8). Changing the email address will only affect the contact email. The email used for logging in will always remain the same. The users have 3 possibilities when sharing their contact information. They can share both their email and phone number or only one. It is recommended to provide at least one form of communication, otherwise users might have a hard time contacting each other. The layout of the settings view is visible in figure 7.


9:20

Search Errands

something
0.00 km 25 EUR 1 min
DETAILS LOCATION RUN

Dog walking
8999.96 km 3 EUR 5 min
DETAILS LOCATION RUN

new
0.00 km 15 EUR
DETAILS LOCATION RUN

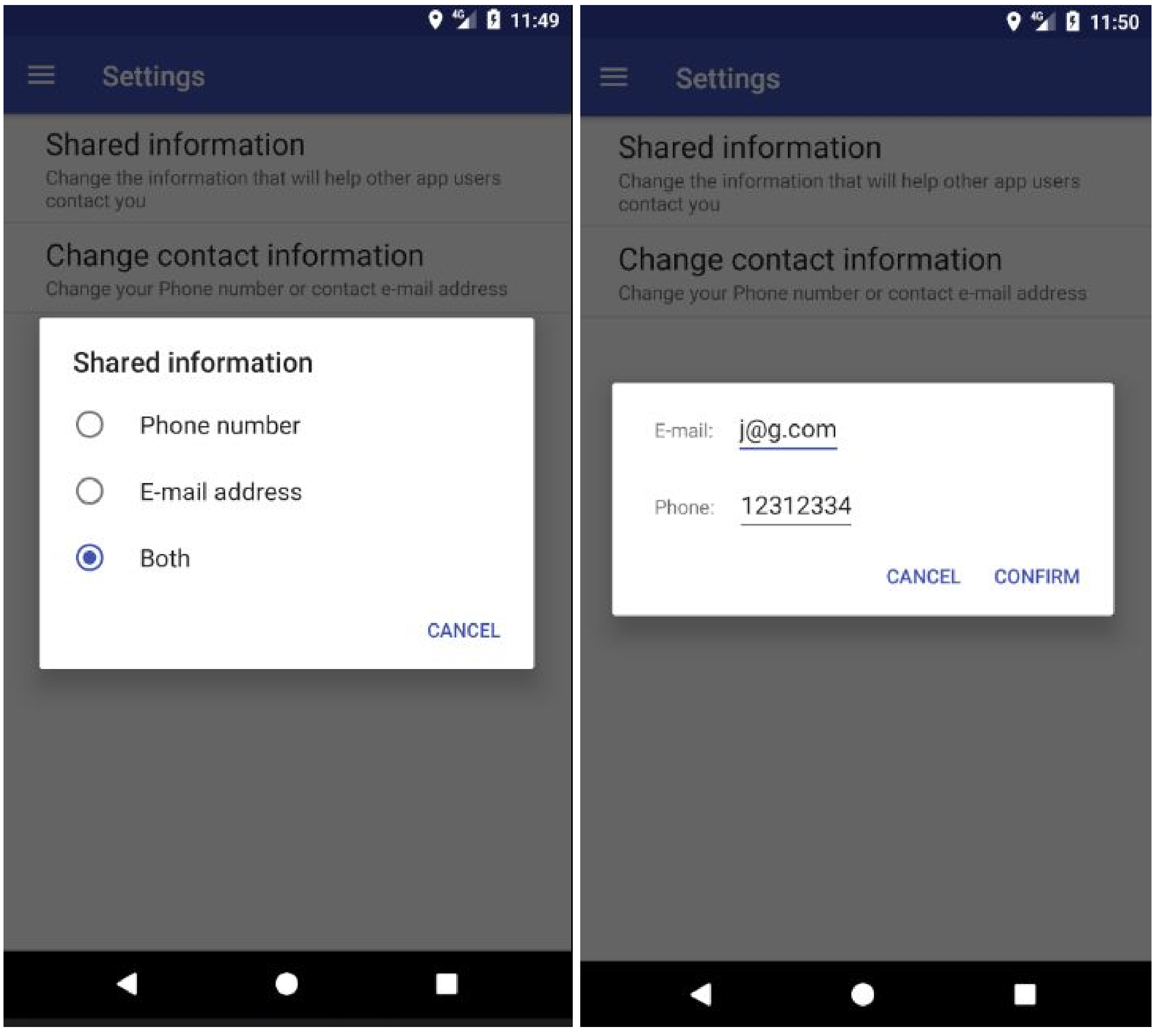
9:22

Settings

Shared information
Change the information that will help other app users contact you

Change contact information
Change your Phone number or contact e-mail address

Figure 7: Search results view on the left and settings view on the right.


11:49

Settings

Shared information

Change the information that will help other app users contact you

Change contact information

Change your Phone number or contact e-mail address

Shared information

○ Phone number

○ E-mail address

[✓ Both]

CANCEL

11:50

Settings

Shared information

Change the information that will help other app users contact you

Change contact information

Change your Phone number or contact e-mail address

E-mail: j@g.com

Phone: 12312334

CANCEL CONFIRM

Figure 8. Shared information configuration on the left and contact information configuration on the right.

Other than the views covered, there are also messages displayed in the form of *Toasts*, *Dialogs* and error messages. Some are displayed when a user tries to perform an action that is not allowed or not available. For example, trying to add an errand without network access. Some are displayed to avoid accidents. For example, accidentally touching the remove button in the added errands view. Some are displayed to ask for specific information, such as the user's phone number if the user logged in with Google.

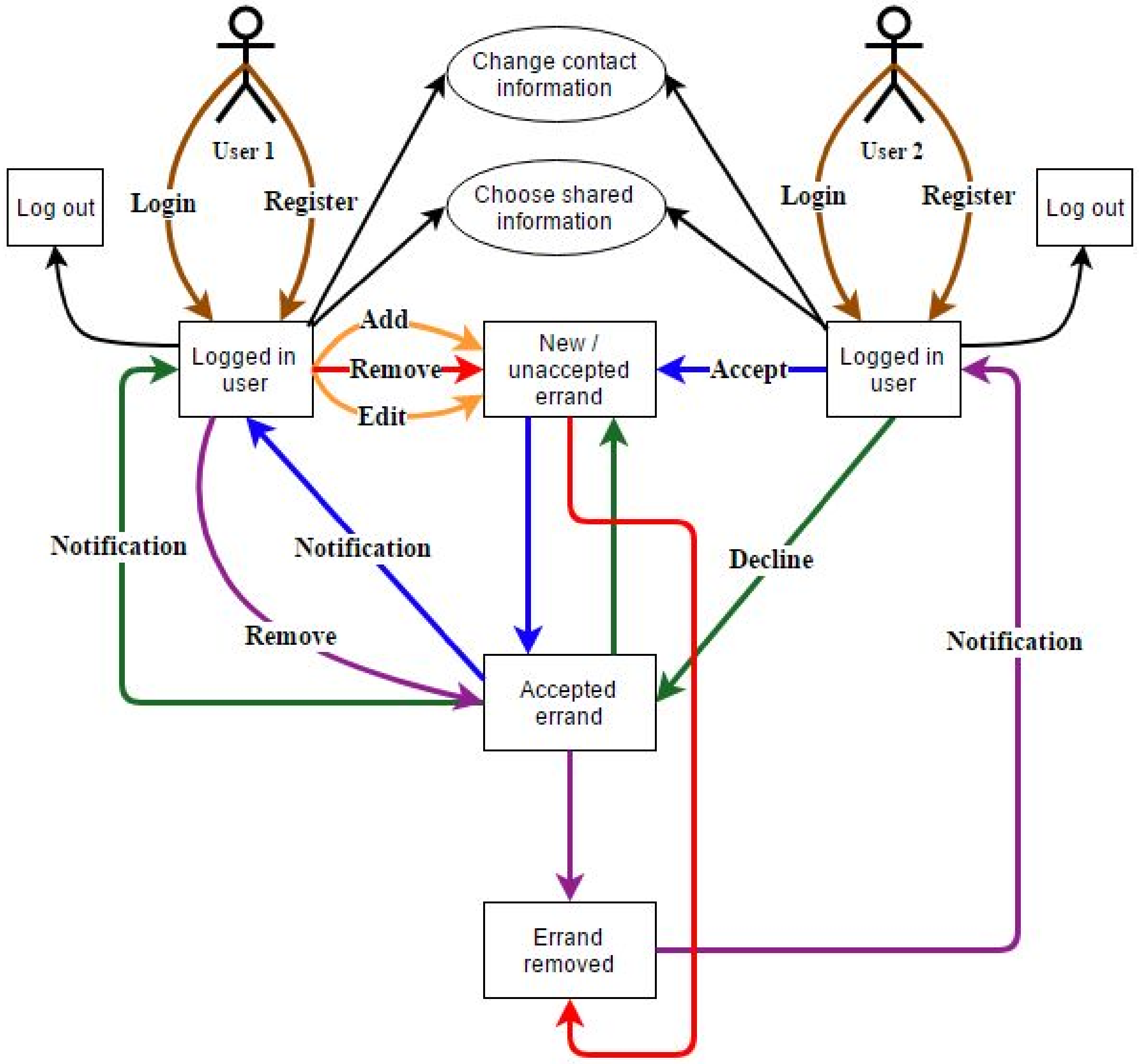


Figure 9. A diagram that illustrates the use of ErrandMe.

Figure 9 demonstrates the use of ErrandMe from the view of an errand provider and a person looking for errands. The figure also displays the life cycle and possible state changes of an errand. The arrows with the same color demonstrate the changes related to one specific operation initiated by one of the users.

3.2.4 Architecture

ErrandMe consists of two components: the application itself and Firebase. As can be seen from figure 10, only Firebase Authentication and Firebase Realtime Database are used.

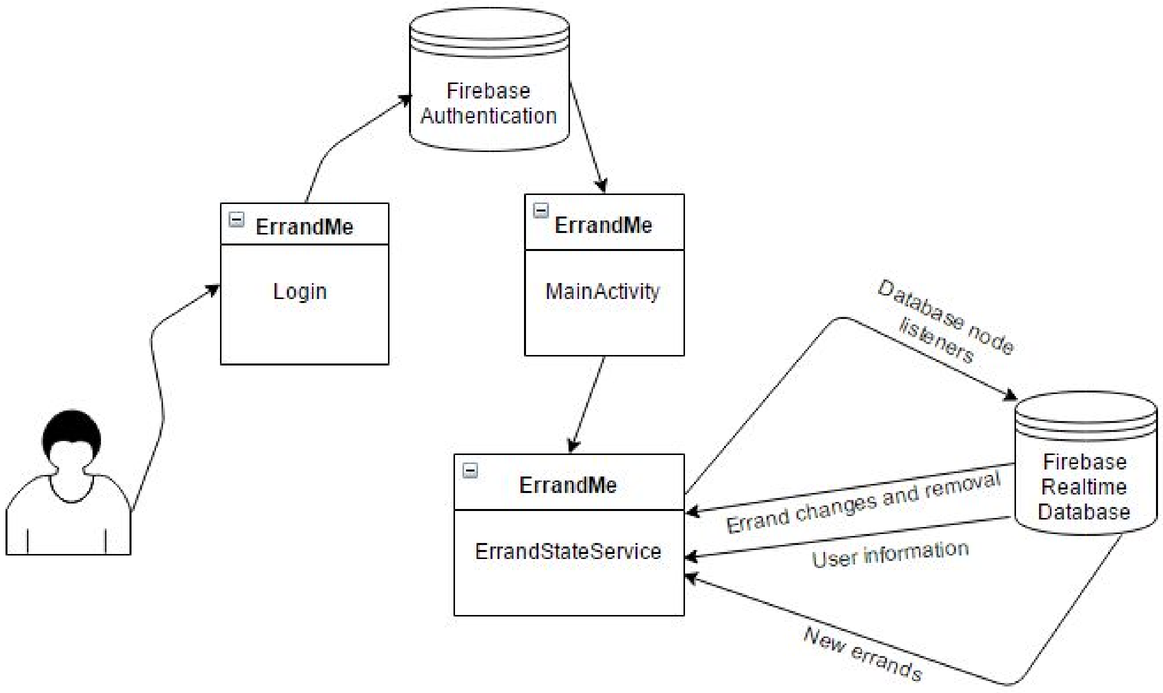
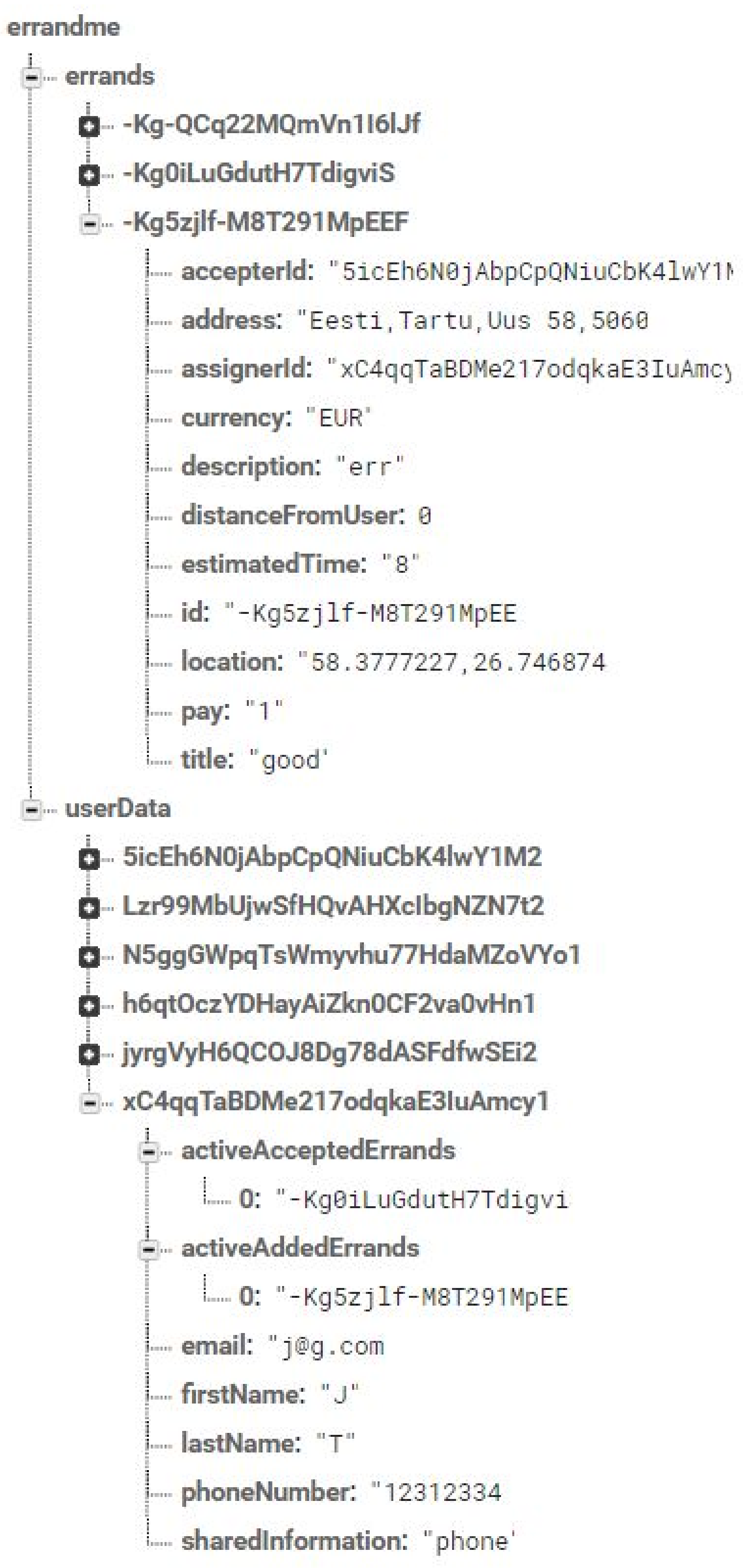


Figure 10. Communication between the application and Firebase.

Figure 10 only displays how the application receives errand and user information updates from the database. In reality, there are many classes that communicate with the database. The classes are discussed in more detail in the methodology subsection.

Firebase Authentication provides different methods to register accounts. It is possible to create email and password based accounts, use the service of different identity providers, but also use custom methods designed by application developers [21].
Firebase Realtime Database is different from databases such as MongoDB and SQLite. Everything is stored as JSON, and it is possible to store and retrieve whole Java objects without losing any information [22]. The data has a tree-like structure, but with one exception: there cannot be nodes without any children.
ErrandMe uses the Java object storing functionality to the fullest, everything is stored and retrieved as an object. As can be seen from figure 11, there are only objects of 2 different classes stored in the database. That is because ErrandMe has only 2 base classes, an *Errand* class and a *User* class.



```
errandme
  errands
    -Kg-QCq22MQmVn1I6lJf
    -Kg0iLuGdutH7TdigviS
    -Kg5zjlf-M8T291MpEEF
      accepterId: "5icEh6N0jAbpCpQNiuCbK4lwY1
      address: "Eesti,Tartu,Uus 58,5060
      assignerId: "xC4qqTaBDMe217odqkaE3IuAmc
      currency: "EUR"
      description: "err"
      distanceFromUser: 0
      estimatedTime: "8"
      id: "-Kg5zjlf-M8T291MpEE
      location: "58.3777227,26.746874
      pay: "1"
      title: "good"
  userData
    5icEh6N0jAbpCpQNiuCbK4lwY1M2
    Lzr99MbUjwSfHQvAHXclbgNZN7t2
    N5ggGWpqTsWmyvhu77HdaMZoVYo1
    h6qtOczYDHayAiZkn0CF2va0vHn1
    jyrgVyH6QCOJ8Dg78dASFdfwSEi2
    xC4qqTaBDMe217odqkaE3luAmcy1
      activeAcceptedErrands
        0: "-Kg0iLuGdutH7Tdigvi
      activeAddedErrands
        0: "-Kg5zjlf-M8T291MpEE
      email: "j@g.com
      firstName: "J"
      lastName: "T"
      phoneNumber: "12312334
      sharedInformation: "phone"
```

Figure 11. The database of ErrandMe.

The user accounts are not stored in the Firebase Realtime Database, but under Firebase Authentication. Depending from the application, there might be no need to store any user information in the database at all. ErrandMe, however, needs more than just email addresses and passwords. That is why user information is also stored in the database.

3.3 Methodology

ErrandMe is divided between the following packages:

1) activities
2) adapters
3) baseclasses
4) fragments
5) other
6) receivers
7) services

3.3.1 Activities

The activities package consists of the following *Activities*:

- *MainActivity*
- *LoginActiviy*
- *RegisterActivity*

RegisterActivity is meant for creating user accounts. The accounts are created by using Firebase's email and password based authentication. Once an account is created, a new *User* object is added to the database under the UserData node to represent the created user.

LoginActivity is meant for handling user login. In this activity, both the email based login and Google sign-in are handled.

MainActivity is one of the most important classes in the entire application. It is responsible for obtaining and updating the device's location and setting up all the necessary services and receivers. All of the fragments in the fragments package are embedded in this activity. The *MainActivity* is called after a successful login with an existing account or after bringing ErrandMe to the foreground. For obtaining location, *MainActivity* uses the Google Location Services API, which is a part of Google Play services. There are several options when using the API. For example, it is possible to choose between accurate data or battery efficiency when obtaining the user's location. ErrandMe prioritises data accuracy, although the obtained location can still be inaccurate. Inaccuracy can be caused by different factors. Some location determination methods might not be accurate indoors, and some methods might only be accurate in certain scenarios.

3.3.2 Adapters

The adapters package consists of the following *BaseAdapters*:

- *AcceptedErrandsAdapter*
- *AddedErrandsAdapter*
- *ErrandsPagerAdapter*
- *SearchResultsListAdapter*

AcceptedErrandsAdapter is an adapter that contains the user accepted active errands. It is used to display all of the accepted errands in the accepted errands view. In this class, all of the accepted errand related activities (viewing details, contact information, errand location on a map and declining an errand) are handled with *setOnClickListener* methods.

AddedErrandsAdapter is an adapter that contains the user added active errands and is used to display all of the added errands in the added errands view. In this class, all of the added errand related activities (viewing details, contact information of an accepted errand, editing an errand and removing an errand) are handled with *setOnClickListener* methods.

ErrandsPagerAdapter is an adapter for swiping between the *AddedErrandsAdapter* and the *AcceptedErrandsAdapter*.

SearchResultsListAdapter is an adapter that holds all of the errands that match a user's errand search query. In this class, all of the necessary methods (viewing an errand's details, location on a map and running an errand) are handled with *setOnClickListener* methods. The class also contains a method for updating the errands that are displayed in the search results view when errands have been edited. If an errand is edited in a way that it does not match the user's search query anymore, it is removed from the adapter and the search results view immediately.

3.3.3 Baseclasses

The baseclasses package contains the 2 base classes of Errandme: the *User* class and the *Errand* class.

User class is used to represent the users of the application. The following user related information is held in the *User* class:

1) name
2) phone number
3) email address
4) shared information
5) active added errands
6) active accepted errands

Name, phone number and email address are necessary as means of communication and trustworthiness. The email address stored in the user class is not exactly the same as the one used for logging in. It is only the contact email. Shared information specifies the information that a user shares with others. It can be be the user's phone number, email or both. The active added errands and active accepted errands are *ArrayLists* of errand identificators with added and accepted errands correspondingly. Storing the added and accepted errand in *ArrayLists* makes it possible to obtain a user's errands faster when listening for initial information from the database.

Errand class is used to represent the errands. The following information is stored in the class:

1) errand identificator
2) errand assigner identificator

3) errand accepter identificator
4) title
5) description
6) location
7) address
8) pay
9) currency
10) estimated completion time
11) distance

The title, description and pay values have limitations. The title of an errand cannot be more than 200 characters long, and the description has a limit of 2000 characters. The pay must be less than 20 digits long. The location is a comma separated value that consists of the latitude and longitude of the errand doing location. It is used to measure the exact distance between the errand accepter and the errand and to display the errand on a map. The address is an exact address of the location where the errand has to be completed. Pay and currency specify the pay that a person receives for completing an errand. The estimated completion time has two possible values: none or a rough estimate of the errand completion time. With some errand it might be hard to estimate the completion time, hence the user may choose not to add it at all. The distance value is only necessary for users who have accepted an errand, and it contains the distance between the user and the errand doing location in kilometers.

3.3.4 Fragments

The fragments package contains the following *Fragments*:

1) ActiveErrandFragment
2) AddFragment
3) ChangeInformationDialogFragment
4) ContactInfoFragment
5) DetailsDialogFragment
6) EditiDialogFragment
7) LocationDialogFragment
8) MyErrandsFragment
9) PersonalInfoDialogFragment
10) SearchFragment
11) SearchResultsFragment
12) SettingsFragment

ActiveErrandFragment is a fragment that fills the *ListView* that is displayed to the user in My Errands view with the correct content. The content of the *ListView* depends on which errands are visible. The *ListView* contains the user's added errands if the added errands view is open, and it contains the accepted errands if the accepted errands view is open.

AddFragment is a fragment that handles adding new errands and updating added errands when edited by the user. For manually entered errand addresses, the fragment uses the *Geocoder* class, which makes it possible to transform an address into latitude and longitude values and vice versa. The fragment also performs valid input checks before allowing the user to add an errand.

ChangeInformationDialogFragment is a *DialogFragment* for changing a user’s contact information (email and phone number).

ContactInfoFragment is a *DialogFragment* that displays the contact information of a person who added or accepted an errand.

DetailsDialogFragment is a *DialogFragment* that displays all the necessary errand details to the user.

EditDialogFragment is a *DialogFragment* that is displayed to the user when clicking the edit button in the active added errands view. The fragment uses the same layout as the *AddFragment* and performs valid input checking as well.

LocationDialogFragment is a *FragmentActivity* that displays the location of an errand on a map. The map is obtained by using the *GoogleMaps* class together with the *OnMapReadyCallback* method. The map is added as a *SupportMapFragment* in the fragment’s layout.

MyErrandsFragment is a fragment that sets the adapter (*ErrandsPagerAdapter*) for the *ViewPager* that is used to switch between the user’s active added errands and active accepted errands.

PersonalInfoDialogFragment is a fragment that enables the user to change his or her contact information (phone number and email).

SearchFragment is a fragment that displays the errand search view. When the user presses the search button, the fragment checks if everything entered is valid and the task of filtering errands based on the search query is passed on to the *SearchResultsFragment*.

SearchResultsFragment is a fragment that filters errands after an errand search query has been made by the user. The errands that do not match the user’s search query are filtered out step by step according to the checkboxes and the values that the user entered in the *SearchFragment*. The errands are filtered out from all of the errands that are in the database.

SettingsFragment is a fragment that allows a user to configure the available settings. The fragment allows a user to open a dialog to change contact information or change what information is shared with other users.

3.3.5 Other

The other package contains a class named *FireBaseMultiQuery*.

FireBaseMultiQuery is a class for applying several Firebase *singleValueEventListeners* on different database nodes. The purpose of the class is to receive the exact time when all of the listeners have obtained their values, making it possible to manipulate the data obtained from the listeners all at once. *FirebaseMultiQuery* uses the *Task* class, which is used to perform an asynchronous operation, and it is provided by Google Play Services.

3.3.6 Receivers

The receivers package contains a *BroadcastReceiver* named *BootReceiver*.

BootReceiver is a receiver that detects when the device is booted. On device boot the *onReceive* method is called, and the user is logged out of the application.

3.3.7 Services

The services package contains an *IntentService* named *ErrandStateService*.

ErrandStateService is an *IntentService* that listens for changes in the userData and the errands node of the database. This service is responsible for receiving and renewing almost all of the user and errand related data that is used within the application. Within this class, a Firebase *ValueEventListener* is attached to the user's corresponding userData node and a *ChildEventListener* is attached to the errands node. When new errands are added or existing errands are edited, all of the changes that have to be made in the application are taken care of in this class. *ErrandStateService* is also responsible for sending notifications to the user when his or her errands have been accepted, declined or removed. As the class is a service, it does not stop working when the application is in the background, but instead continues to update all of the data in the application and sends notifications to the user when necessary.

There are also some general rules that are followed in the whole application. When performing any operations that could cause data corruption and concurrent modification issues, Firebase *Transactions* are used. *Transactions* make it possible to update database data without worrying about concurrent data modification issues. For example, the errand owner trying to remove an errand and some other person trying to accept the errand at the same time.
As the application requires network connection to work, network connectivity is always checked before performing any operations that require having it, and the user is notified if network is not available. As location is also a crucial part of the application, the user is strongly recommended to enable location on his or her device if it is not available. Trying to use the device's location without location being enabled will result in a dialog that allows the user to navigate to the Location settings. Devices with API 23 and above also require the application to ask for permissions to use some features of a device. ErrandMe requires permission to use the user's location and if the user declines the permission, it will be asked for again when trying to use features that require location to be enabled.

3.4 Conclusion

ErrandMe, an application made with the use of Firebase, makes it possible to find and provide errands in real time. People can provide and search for errands of any kind, pay and duration. Users are notified when their errands are accepted, declined or removed, taking away the need to constantly check the application for updates. Not only can the users manipulate with errands, but also change their contact information and choose what private information to share with other users. By providing users a chance to use their location to add and search for errands, ErrandMe makes it simple to add and search for errands even without knowing an address.

4 Results and analysis

4.1 Introduction

This chapter analyses the performance and validity of the non-functional requirements of ErrandMe. First, the query speeds of the most important functionalities of ErrandMe are covered. Secondly, the non-functional requirements are verified through tests with real users. Last but not least, the deficiencies and user opinions of ErrandMe are covered.

4.2 Performance tests

The performance of ErrandMe was analysed through query speed tests. The time it takes to add certain data to the database and receive the updated data in the listeners used in ErrandMe was measured. The following times were measured:

- The time it takes for ErrandMe to receive the user's data and the first errands after logging in;
- The time it takes for the main errands node listener and the user's node listener to receive updates when errands are added, removed, accepted and declined.

The tests were performed on a Sony Xperia Z5 Compat E5823 with Android 7.0. Every test was performed 3 times, and an average of the tests was taken. The tests were performed using the *currentTimeMillis* method of the *System* class.

The start time was stored in a varible:

long startTime = System.currentTimeMillis();

In the node listeners in ErrandStateService where the modified data reaches after database changes, the start time was subtracted from the current time:

long totalTime = System.currentTimeMillis() - startTime;

4.2.1 Initial data

The purpose of the initial data receiving test was to measure how long it takes for the errands and the user's node listener to receive initial data from the database after logging in with an existing account. The time measuring was started in the *OnCreate* method of *MainActivity*.

4.2.2 Adding a new errand

With this test, the time it takes to add a new errand was measured. The time measuring started from the moment the user pressed the "CONFIRM" button in the errand adding view. The test was done separately for both the data that reaches the user's node listener and the errands node listener. The user's node listener receives an updated *User* object with a new errand in the *userAddedErrands* variable, and the errands node listener receives the newly added errand.

4.2.3 Editing an errand

The purpose of this test was to measure how long it takes the errands node listener to receive updated data after editing an errand. The user's node listener does not receive any updates when editing an errand, which is why no measurements related to the user's node had to be made. The test was performed the same way as the new errand adding test, but from the errand editing view.

4.2.4 Removing an errand

This test was performed to measure how long it takes the errands node listener and the user's node listener to receive updated data after removing an errand from the active added errands view. The time measuring started after confirming the removal of an errand.

4.2.5 Accepting an errand

This test was done to measure the time it takes to receive updated errand and user data after accepting an errand. The time measuring began after pressing the "RUN" button of an errand in the search results view.

4.2.6 Declining an errand

With this test, the time it takes to receive updated errand and user data after declining an errand was measured. The time measuring began after confirming the declining of an errand.

4.2.7 Test results

Table 1. Query speed testing results in milliseconds.

Tests	Initial data	Adding	Editing	Removing	Accepting	Declining
Errands node	2267 ms	292 ms	352 ms	50 ms	15 ms	76 ms
User's node	2360 ms	296 ms	-	68 ms	28 ms	56 ms

As can be seen from table 1, the initial data receiving time was the longest, followed by errand editing and adding. It took around 2 seconds to receive the initial user and errands data from the database, which means that the user has to wait for that long to see his or her added and accepted errands after logging in. Errand accepting, declining and removing all took less than 100 ms on average. That can be explained by these actions not requiring many operations or checks to be performed.

4.3 User tests

To check the validity of the non-functional requirements set in chapter 2, the author performed tests with 5 different people. All of the testers have prior experience using Android applications and own Android smartphones, which were used for performing the tests. The smartphones used included devices with older Android versions (Android version 4.0.3 and 4.4) and devices with Android 7.0. The testers were all in the age group 20–25.

To test the non-functional requirement, which required at least 4 people out of 5 being able to perform certain tasks, the following had to be done by each tester:

1) Opening the application and navigating to the registration view;
2) Registering a new account;
3) Adding one new errand with the use of the device's location;
4) Confirming that the errand was successfully added (navigating to the My Errands view);
5) Viewing the added errand's details;
6) Changing the location of the added errand to Estonia,Tartu and confirming that the location changed;
7) Removing the added errand;
8) Searching for an errand with the word "walking" in the title within 2 minutes, checking its location on a map in the search results view and accepting the errand:
9) Confirming that the errand was accepted (navigating to the My Errands view);
10) Declining the accepted errand;
11) Changing contact information;
12) Changing what information is shared with other users.

All of the 5 testers managed to perform all of the tests without any real difficulties. Only the searching test was somewhat difficult for one user. 4 users out of 5 managed to search for the errand with "walking" in the title in less than a minute, and one user took 4 minutes to find the errand. The tester did not notice the error messages, which were not visible on the screen as the view was scrolled down. Other users tried scrolling up immediately when some fields were not filled correctly, which made them notice the messages faster. No assistance was required with any of the tasks, and the tasks were provided to the users without specifying the location of the functionalities in the application.

4.4 User opinions

The same 5 people who tested the application were also asked to provide their opinion on the application. They were told to specify the issues and deficiencies they saw, present some ideas for improvements and also talk about what they liked.

The testers brought up different issues and ideas for improvements. 2 out of 5 users thought that error messages should be handled in a different way in views where the user has to scroll. They thought that the application should move directly to the fields with error messages instead of staying still. 2 testers also thought that the user should instantly be directed to the My Errands view after adding a new errand. These were useful recommendations and the changes were made in the application.
Following are the issues and suggestions that were brought up by only one user:

- The design of the navigation menu could be better;
- The buttons in the My Errands and the search results view should be bigger;
- There should be a way to see a history of previously added and accepted errands;
- There should be a way to communicate with other users in the application.

The design issue is a question of personal taste. As only one user considered the design as a weak point, and other testers liked the design of ErrandMe, it will not be changed. Chatting will be added to the application in the future, and the history viewing functionality will also be looked into.

As mentioned, 4 out of 5 testers liked the design of the application, especially how simple it is to navigate between different functionalities. All of the testers liked the idea of ErrandMe and thought that it could be quite useful. 3 users also mentioned that the possibility to use the current location and view the errands on a map are excellent features.

4.5 Conclusion

Other than receiving the initial user and errands data, information moves between the database and ErrandMe quite fast, and there do not seem to be any real issues. ErrandMe is an application that people do not seem to have a hard time with, the intuitiveness of learning to use different functionalities and navigating between different views contributing to it. Users are generally satisfied with what functionalities are already available and how they have been implemented. Nonetheless, there are still improvements that can be made and that need to be made to make ErrandMe as user-friendly, efficient and usable as possible.

5 Conclusion and future work

As a result of the thesis, an Android application called ErrandMe was developed. ErrandMe provides an intuitive environment for advertising errands and finding people to run errands for a monetary reward. It is possible to add, edit and search for errands, all of which allow the use of the device's location and other criteria. All errands have a title, description, pay, currency and a location. Some errands have an estimated completion time. Errands can be accepted, declined and removed, all of which result in a notification being sent to the errand provider or errand accepter. Although it is not possible to chat within the application, contact information of an other user can be viewed when two users share an errand (one as the owner and other as the accepter). It is also possible to choose and change the contact information that is shared.
Compared to similar existing solutions, the created application treats all users equally as anyone can accept an available errand and prior experience is not something that provides an advantage when looking for errands.

Like with most mobile applications and software development projects in general, it is not a quick process to create a fully finished product; especially if there is only one developer involved. The application created was no exception; some of the issues will have to be solved in the future to make ErrandMe as efficient and user-friendly as possible.
Without a system that manages worker registration and verification, ErrandMe cannot be used in countries where that is required. Without supporting several languages people who do not speak English will not be able to use the application; and due to using a free plan of Firebase, problems will occur if the user base starts growing.
User security can also be an issue as there are many malicious people and frauds. For example, there might be some users that intentionally do not come to the errand location after accepting an errand, even if they promised to come at a certain time.
As a future project a worker registration system will be implemented and more languages will be added. Issues caused by using the free plan of Firebase will be solved by purchasing a premium plan; this will only be necessary if the application has a sufficient and steadily growing user base. For handling fraud, thorough investigation has to be done to analyse different possibilities and to ensure that it is not possible to bypass the security measures.

6 References

[1] Maksu- ja Tolliamet. Töötamise registreerimine lihtsustatud korras. https://www.emta.ee/et/ariklient/registreerimine-ettevotlus/tootamine/tootamise-registreerimine-lihtsustatud-korras (28.04.17)
[2] Android Developers. Dashboards. https://developer.android.com/about/dashboards/index.html (19.02.2017)
[3] Firebase. Pricing. https://firebase.google.com/pricing/ (30.04.2017)
[4] Samsung. http://www.samsung.com/global/galaxy/galaxy-s8/specs/ (25.04.2017)
[5] CNET. Best phones of 2017. https://www.cnet.com/topics/phones/best-phones/ (25.04.2017)
[6] Android. https://www.android.com/ (28.03.2017)
[7] IDC. http://www.idc.com/promo/smartphone-market-share/os (19.02.2017)
[8] Android Developers. https://developer.android.com/studio/features.html (19.02.2017)
[9] Android Developers. ADT Plugin (DEPRECATED). https://developer.android.com/studio/tools/sdk/eclipse-adt.html (19.02.2017)
[10] Gosling J., Joy B., Steele G., Bracha G., Buckley A. The Java Language Specification. Java SE 8 Edition. New Jersey: Pearson Education. 2014
[11] World Wide Web Consortium. Extensible Markup Language (XML) 1.1. https://www.w3.org/TR/2004/REC-xml11-20040204/ (29.04.2017)
[12] Muschko B. Gradle in action. New York: Manning Publications. 2014
[13] Kaplan E., Hegarty C. Understanding GPS: Principles and Applications. Second Edition. Boston: Artech House. 2006
[14] Android Developers. Overview of Google Play Services. https://developers.google.com/android/guides/overview (11.03.2017)
[15] Firebase. https://firebase.google.com/ (26.04.2017)
[16] Kumulos. https://www.kumulos.com/ (09.05.2017)
[17] Kinvey. https://www.kinvey.com/ (09.05.2017)
[18] GoWorkaBit. https://goworkabit.com/mis-on-tooamps-ja-muud-kusimused (28.03.2017)
[19] TaskRabbit. https://support.taskrabbit.com/hc/en-us (28.03.2017)
[20] AgentAnything. http://www.agentanything.com/how-it-works/ (28.03.2017)
[21] Firebase. Firebase Authentication. https://firebase.google.com/docs/auth/ (29.04.2017)
[22] Firebase. Firebase Realtime Database. https://firebase.google.com/docs/database/ (29.04.2017)

Appendix

I. The application

The source code can be viewed, downloaded and cloned from GitHub:
https://github.com/joosep41/ErrandMe

The following requirements must be met before installing and using ErrandMe:

1. A smartphone with Android version 4.0.3 or higher;
2. Access to Google Play services;
3. The device must allow installing applications from unknown sources.

Allowing unknown sources:

- Open the device's Settings application;
- Navigate to security related settings (usually under "Security");
- Find and enable the settings that allow installing applications from unknown sources.

To use ErrandMe, the following has to be done:

- Download the .apk file from GitHub https://github.com/joosep41/ErrandMe/releases/download/1.0.0/ErrandMe.apk;
- Get the downloaded file on an Android smartphone (if not already);
- Locate the downloaded file by using a file managing application or some other tool, depending on the applications available on the device;
- Install the application.

II. License