ProntAPP: A mobile question answering application for medicines

Luis Miguel Mancilha de Almeida Martins de Jesus
luisjesus@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal
November 2015

Abstract
We have been witnessing an increased amount of medical data that is available online. Such data is accessed not only by medical staff but also by patients, and the public in general. However, with that much data, it has become increasingly difficult to search for the correct answers we are looking for in a specific situation. ProntAPP is an Android and iPhone mobile application that aims at fulfilling this gap with a question-answering system in Portuguese for medicines. In this document, we describe the main features of ProntApp, we present the current state-of-the-art of the medical domain question-answering systems, web-based systems and mobile applications for medicines, its architecture and the implementation decisions. Finally, we describe the experimental evaluation that we conducted to validate ProntApp and the results obtained, and draw some final conclusions.

Keywords: mobile application, question-answering system, medicines, voice recognition

1. Introduction
More and more medical information is becoming available on the web (e.g., Medscape\(^1\), Epocrates\(^2\), etc.). Both medical staff and common users need to access this information efficiently. However, with that much information, it has become increasingly difficult to search for the correct answers we are looking for in a specific situation. For example, when a user searches on Google "What are the contra-indications of Vastarel?", the answer consists of a list of search hits, meaning that the answer is not answered immediately. There are several web-based medical systems that present detailed medicine information for certain medical questions (e.g., Infarmed\(^3\), Medscape). However the questions are usually keyword-based, therefore they must be posed with limited search terms so that the system is able to answer. Medical professionals are used to search for medicines with very specific keywords and terms. Contrarily, the common user is not that familiar with such terms. In addition, the common user may not be a computer expert, and may have some difficulties using such keyword-based interfaces. These interfaces should be available for users in a more convenient and ever-present way, taking advantage of the new technologies and gadgets, and therefore enabling more users to access the information they need, exactly when it is convenient for them.

Currently, the most popular gadget in the world is clearly the smartphone; as a matter of fact, such devices are becoming more and more popular. It is estimated that, by 2017, at least 34% of the world’s population will own a smartphone, a fast increase with respect to 9.6% in 2011\(^4\). In terms of the Portuguese market, 55% of the country’s entire population owns a smartphone, \(^5\) and this number is rapidly increasing. We believe that it is of utmost relevance to have information about medicines available on a smartphone. This enables to offer access to such information anywhere, in a convenient way, and taking advantage of many of the smartphone’s unique features.

1.1. Objective
The main goal of this thesis is to develop a mobile application, named ProntAPP, to answer questions about medicines in Portuguese. This application is intended to be used by the common user, including elder people, as well as medical staff. However, elder people are not the most proficient smartphone users. Additionally, we had the opportunity to gather some feedback with some medical doctors to find out how they would use such an application, and the most important features that the application should have, in addition to answering medical questions. From their feedback, we came to the con-

\(^1\)http://www.medscape.com/
\(^2\)http://www.epocrates.com/
\(^3\)http://www.infarmed.pt/
clusion that medical doctors deal with the same set of medicines on a regular basis, and would benefit from having them easily and rapidly accessible in the application. In addition, they proposed that, since the application is also targeted to the common user and elder people, there should be some kind of reminder to take the medicines in due time, with options to control the type of dosage, amount, and several hours of the day.

1.2. Requirements
Taking the target users and their limitations into account, and also the feedback that we gathered from medical doctors, the application must fulfill the following requirements:

- **R1.** Users must be able to pose natural language questions about medicines in Portuguese, and thus should not be limited to keyword-based questions.
- **R2.** The information about medicines that is returned as an answer must be as detailed as possible.
- **R3.** Users must be able to search for what they need with a simple interface (e.g., using a speech recognition interface), and in a small amount of steps.
- **R4.** The application should provide easy access to frequently used medicines.
- **R5.** The application should remind the user to take their medicines in due time with the ability to choose the type of dosage, amount, and several times a day.
- **R6.** The application must be available in a large number of mobile devices, ranging from low-end to high-end devices, and running different operating systems.

Currently, medical mobile applications available from Google Play, and the App Store do not fulfill this set of requirements. In particular, most applications are only available in English, and do not offer a question-answering functionality. In addition, only eMed.pt offers an alarm functionality, however a very basic one (without the ability to set multiple hours per day). We provide more details about such solutions in Chapter 2.

1.3. Proposed Solution
In order to satisfy the set of requirements mentioned above we developed two mobile applications, one for Android and one for iPhone, which present the following set of features:

- **Data about medicines:** ProntAPP provides its users with vast information about medicines, such as contra-indications, pricing and dosage.
- **Question-Answering for medicines in Portuguese:** ProntAPP recognizes Portuguese natural language questions. It also lets users search for the information they need by posing spoken questions to the system, and by searching for keywords.
- **Online server communication:** ProntAPP leverages on the computational power of online servers to perform highly intensive tasks, like voice recognition and natural language processing.
- **Voice recognition:** To further facilitate the user’s input, ProntAPP provides a voice recognition interface.
- **Favorites:** ProntAPP has a favorite medicines list, so they are easily accessible. The user can add a medicine to their favorites list, and also remove it.
- **Alarm:** ProntAPP has the ability to set alarms for medicines, with the ability to choose dosage type, amount and multiple hours per day. This alarm system reminds the user to take his medicines in due time.
- **Large Number of Supported Devices:** The developed mobile applications support 96% of all Android devices\(^6\) and 84.2% of all iPhone devices\(^7\).

1.4. Contributions
As a result of this thesis, two native mobile applications were developed: one for Android and one for iPhone. The applications will be available for download on Google Play and AppStore. The main features of these applications are the ability to answer natural languages in Portuguese about medicines, voice recognition using Google’s voice recognition and Nuance Dragon, the ability to add a medicine as a favorite and the ability to add a repeating alarm for several hours of the day for a given medicine. This application is the only one currently available that has this functionalities, being innovative in its field.

Furthermore, the question-answering module of the application was implemented using MedicineAsk. In order to do this, we had to do some major code refactoring in the online version of MedicineAsk in order to develop an API so that the question-answering functionality could be accessed from outside clients (Android and iPhone).

\(^6\)https://developer.android.com/about/dashboards/index.html  
\(^7\)https://david-smith.org/iosversionstats/
2. Related Work

This section presents several systems that are relevant and related to ProntAPP. First, we compare MedicineAsk[6][3] and MEANS[1], two question-answering systems that support Natural Language questions. Second, we compare Epocrates, Medscape, Drugs.com and eMed.pt which are mobile applications that provide information about medicines. Finally, we present the state of the art in terms of voice recognition systems for mobile applications. In particular, we discuss Google’s voice recognition system. CMU Sphinx, and Nuance Dragon, and draw some conclusions.

2.1. Comparison of question-answering systems

MedicineAsk and MEANS are two question-answering systems for medicines. Both systems are able to provide an answer to Natural Language questions about medicines and active substances. There are some differences between MedicineAsk and MEANS. MedicineAsk uses Infarmed’s website as its information source, while MEANS uses publically available resources. The method used by MedicineAsk for answering the questions is by querying the extracted information using SQL. On the other hand, MEANS annotates the processed documents in RDF[4] to build a graph traversed to answer queries using SPARQL[2]. MedicineAsk can answer questions about medicines, active substances and what they treat. MEANS has a broader scope, answering questions about medicines, active substances and what they treat, who should take medicines and differences between diseases identification. Finally, MEANS supports questions in English, while MedicineAsk supports questions in Portuguese.

2.2. Comparison of voice recognition systems

In this section, we present the state of the art in terms of voice recognition systems. In particular, we discuss Google’s voice recognition system8, CMU Sphinx, developed at Carnegie Mellon University9, and Nuance Dragon10.

Google’s and Nuance’s voice recognition is not open-source, therefore they cannot be modified in any way. CMU Sphinx is open-source with a BSD License, which imposes minimal restrictions on the redistribution of the software. Both Google’s voice recognition system and CMU Sphinx’s are available for free, while Nuance’s is free only up to 20,000 transactions per month, charging $.008 per transaction after that. Google’s voice recognition system and Nuance’s is available for the Portuguese language while CMU Sphinx does not yet support Portuguese language (however the system can be trained). Both Google’s voice recognition and Nuance’s is performed online servers (cloud speech recognition), therefore they are only available if the client is connected to the network, while CMU Sphinx’s voice recognition is performed locally.

2.3. Comparison of existing medical mobile applications

In this section we compare medical mobile applications. In particular, we make a comparison between Epocrates, Medscape, Drugs.com and eMed.pt, which are the mobile versions of their respective web-based counterparts.

All of the above mentioned applications are available for Android and iOS. The eMed.pt application is the exception in terms of language, since it is the only mentioned application with information in in Portuguese. All applications are available to download and use for free, however Epocrates and Medscape have a paid subscription with more functionalities. Offline use is possible in all applications except for Drugs.com which is only available if the device is connected to the network. Keyboard-based search is also supported by all applications, however none of them supports natural language search or speech recognition. Epocrates, Medscape and eMed.pt have a local database on the device which is downloaded from an online server, while Drugs.com does not. Drugs.com and eMed.pt are the only applications developed natively for the operating systems, being developed in Java (native programming language of Android) and Objective-C (native programming language of iOS). Both Epocrates and Medscape are hybrid applications developed in HTML.

2.4. Discussion

Since neither of these applications fulfill the set of requirements listed in Section 1, we propose a mobile application, named ProntAPP, which fulfills them. This application will use MedicineAsk for the question-answering system since it is the most advanced available in Portuguese. In terms of voice recognition, we will use Google’s voice recognition for the Android mobile application and Nuance Dragon voice recognition for the iOS mobile application, since they are the most advanced in their respective operating systems. Although Epocrates, Medscape, Drugs.com and eMed.pt present detailed information about medicines, they fail to meet our requirements of being able to answer natural language questions in Portuguese, having easy access to frequently used medicines, an alarms functionality (except for eMed.pt) and neither of these has a voice recognition system, hence the need for the development of ProntAPP.

8http://developer.android.com/
9http://cmusphinx.sourceforge.net/
10http://developer.nuance.com/
3. Proposed Solution

ProntApp features an online architecture, which means that its main functionalities are only available if the user is connected to the Internet. We have chosen this architecture because it is quite popular in mobile applications nowadays, and further facilitates further ports of the mobile application to other operating systems, since all the main business logic is stored server-side. Furthermore, we can leverage on the computational power of online servers to perform intensive tasks, like voice recognition.

There are two databases in ProntAPP: (1) one available online in the question-answering online server, which we refer to as online database and (2) one local database available offline in the mobile client application, which we refer to as offline database. The online database stores all the medicines data, active substances, interactions, medical conditions, etc. This information is only available in the online database. The offline database is only present in the mobile client application, and only stores the information about medicines added to the favorites list, and about the alarms configured by the user.

The architecture of ProntAPP is represented in Figure 1. ProntAPP works on a mobile phone which runs the mobile client application. The mobile client application is divided into three layers: the Presentation Layer, the Business Layer and the Data Layer. The Presentation Layer provides a user interface where the input of the application is specified (e.g., form inputs, buttons, etc.) and its output is shown (e.g., tables, text, etc.). It also contains the presentation logic which shows the data in the native user interface of the operating system (i.e., in the case of Android, this logic means displaying the data in Android’s views). The Presentation Layer communicates with the Business Layer when a user interacts with the application. The Business Layer contains all the libraries that ProntAPP uses. In particular, it makes the necessary requests to perform the voice recognition (which is performed in online servers), question-answering processing (performed in an online server) and business logic (e.g., alarm functionality, favorites logic, server communication).

When the user accesses its Favorite medicines or the Alarms, the Business Layer requests data from the Data Layer. The Data Layer stores the local database, handles all the database communication using data access components (i.e., SQL Connectors), and converts the data returned by the database into structures that can be handled by the other layers, with database helpers which perform the mapping from SQL cursors into structures.

As mentioned above, an offline database is stored in the mobile client application. Since the search functionality is only available if the user is online, we found that it was not necessary to store a copy of the online database on the client application. Therefore, the local database only stores information about the favorite medicines, not all medicines. When a user adds a medicine as a favorite, all the information is fetched and stored for offline use. The local database also stores information about the alarms scheduled by the user. The online server provides an API that the mobile application uses for the question-answering functionality, via HTTP requests.

4. Implementation

In this section we describe the implementation process of ProntAPP. We start by presenting the chosen target mobile operating systems, Android and iPhone, and our reasons. Next, we present the question-answering module. In particular, the changes made to the MedicineAsk system and the implementation of the API. We proceed by discussing the databases of ProntAPP (online and offline). Next, we describe the implementation of the voice recognition module, and the implementation of the alarms and favorites functionalities. We present the main libraries used in our development process and lastly we present and discuss some screenshots the final user interface.

4.1. Target Mobile Operating Systems

There are three main markets in terms of mobile applications: Android, iOS and Windows Phone. In Portugal, Android and iPhones take the majority of the market share. Therefore, we believe that these are the mobile platforms that must be prioritized. Accordingly, ProntAPP was developed as an Android and an iPhone native mobile application.

---

4.1.1 Android

If we analyze Android versions' market distribution, we find that approximately 99.3% of Android smartphones are on version 2.3.3 (Android SDK 10) or above. Therefore, the application was developed to work on devices with the Android SDK 10 or above. The application is developed in Java language, using Android’s SDK. We use a typical Model-View-Controller [5] software architecture since it is the standard when developing Android mobile applications.

Regarding the voice recognition functionality, ProntApp uses Google’s voice recognition, since it is the most advanced voice recognition system available in Portuguese, and has great support and tools for integrating the system with mobile applications. The user interface is implemented using native Android views, provided by the Android SDK. Furthermore, the voice recognition module is implemented using Google’s Speech Recognizer. The question-answering module is implemented by accessing the new MedicineAsk’s API using an Android software library called RetroFit. In addition, the local database is an SQLite database and is accessed through an Android software library called ActiveAndroid. These software libraries are better described in Section 4.6.1.

4.1.2 iPhone

Approximately 86% of Apple devices are on iOS 8.0 or higher (which means iPhone 4S or better). Therefore, this is the operating system version that most developers target, since they can make use of new features of XCode like Auto-Layout and adapt their design to multiple screen densities. Consequently, ProntApp was developed to work on devices with iOS version 8.0 or higher (including iPads, although the design is not adapted to tablet devices). The application is developed in Objective-C, using the iOS SDK. Like on Android, we use a typical Model-View-Controller software architecture since it is the standard when developing iPhone mobile applications. The iPhone application was developed by the standard iOS design guidelines, meaning that the design was not a direct port from the Android version.

Regarding the voice recognition functionality, ProntApp uses Nuance Dragon Mobile SDK. It is the most advanced voice recognition system available in Portuguese for iPhone. This voice recognition, like Google’s, is processed on online servers.

The user interface is implemented using native iOS views, provided by the UIKit Framework in the iPhone SDK. Furthermore, the voice recognition module is implemented using Nuance Mobile Dragon SDK. The question-answering module is implemented by accessing the new MedicineAsk’s API using an iOS software library called RestKit. In addition, the local database is an SQLite database and is accessed through an iOS software library called FMDB. These software libraries are better described in Section 4.6.2.

4.2. Question-Answering Module

The question-answering module used by ProntAPP is the same used in MedicineAsk. A major part of the development process was the adaptation of MedicineAsk to work in other interfaces, other than simply retrieving HTML. Most mobile operating systems support the display of HTML, however we wanted to have more control over how the data is presented, and not simply display the retrieved HTML, which resulted in the development of an API, so that the mobile client applications can retrieve the response, process it, and display it in its own native user interface, like they were communicating directly with the database.

The MedicineAsk system is very coupled with the web-interface (meaning that the code related to user interface, business logic and database accesses in enter-twined). Therefore, one of the challenges to modifying this Question-Answering module was to separate the presentation logic from the processing logic, so that we could change the data format retrieved by the Question-Answering module.

In particular, an endpoint was exposed through this API called analyze. This endpoint accepts a GET request with a query parameter with the user’s question. After MedicineAsk’s NLI and NLP analysis and database retrieval this endpoints retrieves the data in a JSON format. This JSON response is then parsed on Android and iOS by their respective libraries and displayed on screen using native views.

In addition to the analyze endpoint another endpoint was exposed called alarm_quantity_types which retrieves a JSON response with the alarm quantity types and dosages available for scheduling alarms in ProntAPP’s alarm functionality.

These modifications allowed us to have the same source code of MedicineAsk running on the web (essentially keeping MedicineAsk’s website intact) to also be used by the mobile applications. This approach will also facilitate further updates to the Question-Answering module, since any updates to the MedicineAsk’s NLI and NLP modules are transparently updated and used by the mobile applications, without any need of changes to the mobile applications’ source code.

---

12https://developer.android.com/about/dashboards/index.html
4.3. Database
In this section, we discuss the implementation of both the online database and the offline databases (iPhone and Android).

4.3.1 Online Database
We wanted to enhance the alarms functionality with some features, like choosing the type of medicine dosage and type (e.g. Take 1/2 Pill), therefore we had to extend this database with two new tables called alarm_quantity_type and alarm_quantity_option. The alarm_quantity_type table stores the type of alarm quantities (i.e. pill, applications) while the alarm_quantity_option stores the options for each type (i.e. 1 pill, 1/2 pill, 3/4 pill). The alarm_quantity_option table stores a foreign key to the alarm_quantity_type so that we can infer the options of each type.

4.3.2 Mobile Application Databases
The local database present in both the Android and iPhone mobile applications is in the SQLite format. Although both Android and iPhone provide classes in their SDK to access the database, we chose to use some software libraries. In the case of Android, we chose to use an ORM (Object-relational mapping) library to further facilitate this data access and storage. The software library we chose to use is ActiveAndroid and is further described in Section 4.6.1. The iPhone SDK provides developers with a standard, near-transparent interface to a database file called Core Data. With Core Data, an app can define a database schema, create a database file, and create and manage record data. In the case of iPhone, we chose to interact directly with the database with a software library called FMDB. The reasoning for this choice was the ease of use of FMDB in relation to Core Data (since Core Data is intended to larger projects), since the developer can easily provide their own SQL queries and parse the data from SQL cursors directly. The local database stores all the information relative to the favorite medicines, and also stores information about the alarms.

4.4. Voice Recognition Module
4.4.1 Android
Google provides an API to Android developers to facilitate the integration of mobile applications with its voice recognition system. The Android SDK contains a class called SpeechRecognizer. This class provides access to the speech recognition service, which is an Android service which enables access to the speech recognizer. All the voice recognition is performed on Google’s servers, since it is a heavy task to perform on smartphones, and there is no publicly available SDK announced by Google for using this service offline.

4.4.2 iPhone
For the Voice Recognition Module on iPhone, we chose to use the Nuance Dragon Mobile SDK which features the SpeechKit Framework. The SpeechKit framework is a high-level framework with two major components for developers: the speech recognizer and the text-to-speech synthesizer. For the Voice Recognition Module, we only used the speech recognizer. SpeechKit follows a server-based architecture and relies on the Nuance speech server for voice recognition and text-to-speech synthesis. For voice recognition, the SKRecognizer class sends audio streams to the server which then returns a list of text results.

4.5. Alarms and Favorites
In this section we explain how we implemented both the Alarms and Favorites functionalities in the Android and iPhone applications.

4.5.1 Favorites
When a user is viewing the medicine detail and taps the favorite icon, a query is made to the local database using ActiveAndroid on Android and FMDB on iPhone. This query verifies if the medicine is already present in the favorites table or not. If it is not present, a dialog is presented asking the user to confirm if he wants to add the medicine to the favorites. If the user confirms, a new row is inserted in the favorites table in the SQLite database storing all the information about this medicine. If the medicine is already present in the favorites table, a dialog is presented asking the user to confirm if he wants to remove the medicine from the favorites. If the user confirms the action, the row is deleted from the SQLite database. When the user enters the favorites screen, a query is made to the local database and all the rows of this table are then presented in an Android ListView using a custom Adapter, or in an iPhone UITableView using custom UITableViewCell.

4.5.2 Alarms
When checking a medicine’s information, the user can schedule an alarm by pressing the clock icon. This icon opens a custom dialog. Before the dialog opens, a web-service call is made to the alarm_quantity_types endpoint to retrieve the possible dosage types and amounts. The alarms are scheduled using native alarm managers

---

14http://developer.android.com
(AlarmManager for Android and UILocalNotification for iPhone). Alarms based on these native alarm managers give a way to perform time-based operations outside the lifetime of the application. In our approach to this, when a user schedules, for example, a repeating Monday and Wednesday, two repeating alarms are scheduled with a repeat interval of one week. These native alarm managers themselves do not show the notifications that ProntAPP present, they simply trigger a callback. This means that we had to implement our own callback which fires a notification when it is triggered.

4.6. Software Libraries
In this section we present the main libraries used in the development of the mobile applications and a brief description of all of them, providing insights on their value to mobile software developers.

4.6.1 Android
The software libraries dependencies in Android were managed by using Gradle’s built-in dependency management system. Gradle is an advanced build toolkit that manages dependencies and is integrated into Android’s development IDE, Android Studio. In particular, the libraries used were:

- **Retrofit**: a type-safe REST client for Android built by Square. It was used to communicate with MedicineAsk’s API and retrieve the necessary JSON responses from the server.
- **ActiveAndroid**: an active record style ORM (object relational mapper). It was used to access the local SQLite database containing information about favorite medicines and alarms.
- **Butterknife**: an annotation helper that helps developers to instantiate the views from our activity or fragment. It was used to eliminate all the boilerplate code and make the code simpler and easier to read and maintain.

4.6.2 iPhone
The software libraries dependencies in iPhone were managed by using CocoaPods dependency management system, a dependency manager for Swift and Objective-C Cocoa projects. Although it is not directly integrated into iPhone’s development IDE (XCode) like Gradle is integrated into Android’s development IDE (Android Studio), this is a must have when a developer wants to integrate several different libraries, developed by other users, into their own iOS application. In particular, the libraries used were:

- **RestKit**: a framework for implementing RESTful web services clients on iOS. This library was used to communicate with MedicineAsk’s API and retrieve the necessary JSON responses from the server.
- **FMDB**: an Objective-C wrapper around SQLite. This library is used to access the local SQLite database containing information about favorite medicines and alarms.
- **MLPAutoCompleteTextField**: a library which manages a drop down table of autocomplete suggestions that update as the user types. This library is used to implement the autocomplete functionality when typing a question in the search box.
- **Nuance Dragon Mobile SDK**: a library with features like speech recognition and text-to-speech functionality. We use this library to implement the voice recognition functionality in iOS.
- **ActionSheetPicker**: an iOS library which allows a user to select from a number of immutable options. This is used to present the dosage types and quantities in the alarms section.

4.7. User Interface
In this section, we present and discuss the final user interface of ProntAPP in both the Android and the iPhone versions.

4.7.1 Main Screen
The main screen of the Android application is presented in Figure 2. This screen of the application provides a tab-based navigation throughout the main sections of the application: search, favorites and alarms. On the Android version the tabs appear on the top of the screen and on the iPhone version the tabs appear on the bottom of the screen with icons, following both platforms user design principles and guidelines. In both platforms a help message is presented with instructions to the user, before performing a question. This message is also shown if there are no results for a given search.

Figure 2 presents the user interface after a search. In the blue rectangle with white letters the application notifies the user of the question that was understood. Below that, a list is presented with sections for each active substance and a list of medicines. The results list is clickable and, when a user selects an item from the list, he is presented with the medicine detail screen.
4.7.2 Medicine Detail Screen

The Android’s medicine details screen is presented in Figure 3. The screen presents the user with information about the medicine, in particular it shows the name of the medicine, the laboratory, the price, if the medicine is a generic or not, pharmaceutical form, grouping, compartment and PMU. On the top of the screen, a user can add the medicine to its favorites (or remove it) by pressing the heart icon, or he can add an alarm by pressing the clock icon. When a user clicks the clock icon, he is presented with the add alarm dialog.

There are no particular differences between these screens in Android and in iPhone, other than some recommended spacings. Also, in the iPhone version a back button is present on the screen since it is considered a good practice and iPhones do not have a physical back button. On Android, a user can return to the previous screen by pressing the physical back button of his device.

4.7.3 Add Alarm Dialog

The Android’s add alarm dialog is shown in Figure 4. This dialog is presented when a user clicks the clock icon in the medicine detail screen in order to add a new alarm for a given medicine. In the top of the dialog a user can select the type of dosage (i.e., pill, oral solution, etc.) and the quantity. Next, he can select if the alarm should repeat weekly and days. In the bottom of the dialog, the user can add multiple hours for the alarm.

There are some differences in the implementation and user experience of this screen in Android and in iPhone. In Android, the types of dosage and quantities are presented in dropdown lists called Spinners. In iPhone, there is equivalent for this user control. Therefore, we had to use the software library ActionSheetPicker in order to present a list of selectable options. In addition, we also used this library to select the hours, while in Android the hours are selected through a native DateTimePicker dialog.

4.7.4 Alarms List Screen

The Android’s alarms list screen in shown in Figure 5. This screen is accessible by clicking the Alarms tab in the main screen. In this screen, a list of scheduled alarms is presented. The user can toggle them on or off, and when the alarm is disabled a dialog appears asking the user if he also wants to remove the alarm from the list.

There are no particular differences between these screens in Android and in iPhone, since we were able to use similar user controls in both platforms.
5. Results

This section describes the experiments we conducted to evaluate the ProntApp mobile application. The evaluation was divided into two parts: usability evaluation and server load evaluation. The objective of the usability evaluation is to help us understand our targeted audience and ensure that the application is fulfilling the user’s needs. On the other hand, the objective of the server load evaluation is to ensure that the server is handling the users request properly, and also to detect bugs that may emerged during the development phase.

5.1. Usability Evaluation

In order to evaluate the usability of the developed application, we have collected the feedback from 26 users, ranging from common people to medical staff, which have answered a survey. The questionnaire was organized in such a way that the users fill in the first two sections of the questionnaire (which give general information about gender, age and experience), and then they were asked to interact with the application (iOS or Android) in order to perform three scenarios. No help was given during that time. Finally, the users were asked to fill the third and fourth sections of the questionnaire describing their opinion and impressions regarding the application.

From our 26 users, 96.2% of them stated that they found the application useful and that they would download it to their phone. Only 3.8% of users said that the application was not useful at all and would not download it.

Thirteen users found that the application would be useful in other operating systems. In particular, 72.7% think that Windows Phone would be a suitable platform for ProntAPP, while 18.2% think that ProntAPP should be developed for BlackBerry devices in the future.

Users had some concerns regarding the colors used in the separators, stating that the color contrast is not very good. A bug was found on iPhones 6 that prevented the users from adding an alarm. Furthermore, a user stated that he did not know the meaning of some acronyms, so this should be explained in the application in the help section.

Our tests proved that the features that we added to the application were interesting to our users and proven to be useful. The majority of our users thought that the favorites functionality and the alarms functionality were very useful.

Some users also gave us some insights in terms of features that they want ProntAPP to have in the future. In particular, some users thought that having a map with the location of nearby pharmacies would be very useful.

5.2. Server Evaluation

The tests were performed against the server’s RESTful API. We used the Gatling Stress tool\(^\text{16}\) to perform the load tests.

To achieve reliable feedback, we created a scenario for each of the two different requests: a user performing a natural language search (which we refer to from now on as “natural language scenario”) and a user performing a keyword-based search (which we refer to from now on as “keyword scenario”). In the natural language scenario, the question performed was: What are the contraindications of Paracetamol? (Quais são as contraindicações do Paracetamol), while in the keyword scenario the search as simply Paracetamol. These tests were made with 1 user, 5 simultaneous users and 10 simultaneous users.

The server is hosted on Amazon Web Services’ Elastic Cloud Service (EC2) running on an EC2 small instance with 2GB RAM and 1 vCPU, while the database is hosted on Amazon Web Services’ Redundant Data Storage (RDS) running on an RDS small instance as well.

With 1 user our server handled the requests as we expected. In the natural language scenario, the server took 877ms to answer the request, while it took the server 1969ms to answer the request on the keyword scenario. With 5 requests, we started to notice that the server was not handling concurrent requests properly. This was even more noticeable with ten requests, since the server was simply receiving all the requests and answering all of them at once, basically multiplying the answer time.

A summarized comparison of the average response time can be found on Figure 6, which clearly demonstrates a scalability issue on the server (in red the keyword scenario, in blue the natural language scenario), and an exponential trend more noticeable on the keyword scenario.

Our server load tests detected without a doubt that the server is not handling concurrent requests properly. The causes for this may be many: an unintended database lock while answering a request, a bad implementation of database queries (i.e. mak-

\(^{16}\)http://gatling.io/
ing multiple queries instead of JOIN queries or preloading the queries) or others.

6. Conclusions
In this document, we have proposed the development of ProntAPP, a mobile question-answering system for medicines. We started by presenting the objective, requirements, a brief summary of our proposed solution and contributions. In Section 2, we reviewed the most relevant question-answering systems for medicines, existing Android and iOS medical mobile applications and state-of-the-art of voice recognition systems. Section 3 described ProntAPP's architecture and its main components. In Section 4 we presented the implementation of ProntAPP. And finally, in Section 5 we described the experiments conducted to validate ProntAPP and the results obtained.

As a result of this thesis, two native mobile applications were developed: one for Android and one for iPhone. The main features of this application are the ability to answer natural languages in Portuguese about medicines, voice recognition, the ability to add a medicine as a favorite and the ability to add a repeating alarm for several hours of the day for a given medicine. This application is the only one currently available that has this functionalities, being innovative in its field.

Furthermore, the question-answering module of the application was implemented using MedicineAsk. In order to do this, we had to do some major code refactoring in the online version of MedicineAsk in order to develop an API so that the question-answering functionality could be accessed from outside clients (Android and iPhone).

6.1. Future Work
We have identified some limitations with the mobile applications of ProntAPP and mainly with the online server. In particular, these are the points in which we think ProntAPP could be improved:

- **Server Scalability:** Our stress tests to the server detected that the server is not scaling correctly, which means that it is not handling concurrent requests properly and may be a problem if the applications are used by multiple users.

- **Pharmacies Location:** A feature that would be very valuable to add to the current version of the mobile applications of ProntAPP would be the location of nearby pharmacies.

- **Windows Phone Support:** From our questionnaire we gathered that several people want the application to be available for Windows Phone. This means that a native application in C could be developed for Windows Phone.

- **Tablet Support:** ProntAPP was developed for Android and iPhone smartphones, not taking into account the user interface of tablets. The applications could benefit from having a user interface specifically designed for tablets.

- **User Interface Improvements:** From our questionnaire we gathered some small user improvements that could be made in both versions of ProntAPP, namely: (1) The word "mapped" is not very familiar for a regular user, and should be replaced for a more familiar one; (2) The intention of the icon used to show that an alarm in the alarms is a repeating alarm is not very clear, and therefore it should be replaced; (3) The search box should have a clear button so that a user can delete all the text easily.; (4) The color contrast in the separators is not very clear, and should be changed. and (5) A list of acronyms and their meaning could be added to the help section.

References


