Petri Partanen

Developing Mobile Scent Discrimination Training Application

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Bachelor’s Thesis

22 May 2017

Helsinki
Metropolia
University of Applied Sciences

Author(s) Title Number of Pages Date	Petri Partanen Developing mobile scent discrimination training application 31 pages 22 May 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	
Instructor(s)	Markku Karhu, Principal Lecturer

The goal of this final year project was to develop a prototype of a mobile scent discrimination training application for hobbyists and researchers to train sniffer dogs. The purpose of the application is to provide a well-defined scent discrimination training process and an ability to follow a dog's progress throughout the training. In addition, the application was designed to cater research specific needs, such as capability to select samples used in the training.

The training application was developed for the latest iOS operating system using the Swift programming language utilising common programming design patterns. Furthermore, a networking layer for the application was created using a reactive programming paradigm. The training process was designed based on professional scent discrimination training practices. The scope of the project was limited to the design and implementation of the mobile application.

In conclusion, a functional prototype scent discrimination application for the iOS operating system was created. Further development should include improvements to user-friendliness and usability testing.

Keywords	iOS, Swift, mobile programming

Tekijä Otsikko Sivumäärä Päivämäärä	Petri Partanen Mobiiliohjelmisto koirien hajuerottelukoulutukseen 31 sivua 22.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Information Technology
Suuntautumisvaihtoehto	
Ohjaaja	Yliopettaja Markku Karhu

Insinöörityön tarkoituksena oli suunnitella ja toteuttaa harrastelijoille ja tutkijoille suunnattu prototyyppiversio koirien hajuerottelukoulutukseen tarkoitetusta mobiiliohjelmistosta. Ohjelmiston tavoitteena oli tarjota hyvin määritelty hajuerottelukoulutusprosessi sekä mahdollistaa koirien kehittymisen seuraaminen koulutusprosessin vaiheiden läpi. Tämän lisäksi ohjelmistoon suunniteltiin tutkijoiden tarpeita vastaavia ominaisuuksia, kuten mahdollisuus valita harjoittelussa käytettävät hajunäytteet.

Ohjelmisto kehitettiin viimeisimmälle iOS-käyttöjärjestelmälle käyttäen Swift-ohjelmointikieltä. Ohjelmistoarkkitehtuurin toteutuksessa käytettiin yleisiä suunnittelumalleja ja lisäksi verkkoyhteyksiä hallinnoivat komponentit rakennettiin käyttäen funktionaalista reaktiivista ohjelmointia. Koulutusprosessi suunniteltiin perustuen ammattimaisiin hajuerottelukoulutusmenetelmiin. Insinöörityö rajattiin sisältämään ohjelmiston suunnittelu ja toteuttaminen.

Insinöörityön lopputuloksena syntyi toimiva prototyyppi hajuerottelukoulutusohjelmistosta iOS-käyttöjärjestelmälle. Ohjelmiston tulevan kehityksen tulisi sisältää käytettävyystestauksen tekeminen ja käyttäjäystävällisyyden parantaminen.

Avainsanat	iOS, Swift, mobiiliohjelmointi

Contents

1 Introduction

The project documented in this thesis was initiated in November 2015 when I met a Helsinki University research group in a pet exhibition. The research group was training dogs to smell cancer. I was shocked to hear that such an important research with promising results was battling with financial challenges. I wanted to contribute my time and knowledge for the benefit of the research by developing information systems that could benefit their research.

As a result, the goal of this final year project was to develop a prototype of a mobile scent discrimination training application for dogs. The application was to be used in hobby and research contexts. Hobbyists can use the application to train for example truffle finding dogs. For research use, the application offers more advanced features such as following training progress of multiple dogs, selecting samples used in the training, adding comments to samples and recording whether a dog indicated a sample to contain the target scent or not. For example, a research group could use the application to train multiple dogs to detect cancer scent.

This final year project was limited to documenting the development of the mobile application.

2 Literature Review

This chapter is divided into two parts. The first part consists of a literature review of technical aspects and the latter part analyses related research papers on dog scent discrimination methods used in academic studies.

The literature review gives an overview of technologies used in this project, starting from the targeted mobile device and its operating system. After introducing the target hardware, the chapter further explores the utilised software stack consisting of development environment, programming language, applied design patterns and programming paradigms.

The latter half of the literature review studies several research papers which document how dogs have been trained to smell cancer. The review concentrates particularly on the used training methods, the tools and the stages of the scent discrimination training described in the papers. The last part of the chapter consists of a comparison and summary of the reviewed training methods, tools and stages included in the studies.

2.2 Target Hardware and Operating System

Apple's original iPhone was announced in January 2007, more than ten years later it is the most revenue generating product of Apple with over 50 million units sold in the second fiscal quarter of 2017 generating more than 33 billion revenue for the company [1]. At the time of writing this thesis, the latest iPhone version was iPhone 7 and iPhone 7 Plus, released in September 2016.

Apple's iOS mobile operating system, originally named iPhone OS, was unveiled with the first iPhone in 2007. Since its launch, it has expanded to cover other Apple mobile devices such as iPod and iPad in addition to iPhone. The operating system had its tenth release in September 2016 [2] and as of February 2017, five months after its release, almost 80 percent of the devices running iOS had the latest version of the operating system installed, as can be seen in figure 1.

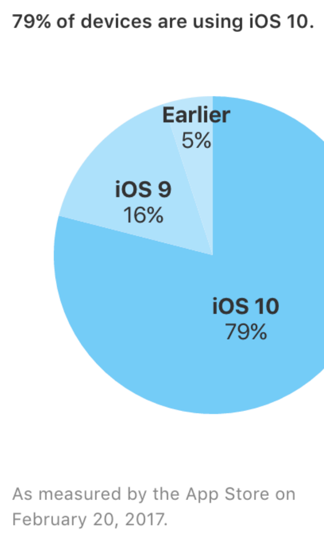


Figure 1. iOS version distribution measured by App Store. Reprinted from Apple (2017) [3]

The second latest release of Apple's mobile operating system had a market share of 16 percent and earlier versions made up five percentage, as seen in figure 1.

When developing the training application during this project, a decision to support only the latest version of the operating system was made as it held the largest market share. The device of choice was iPhone 7 because it was readily available.

2.3 Xcode Integrated Development Environment

Integrated development environment (IDE) is software that provides comprehensive development tools in a single package. It usually combines tools and features such as an automated build mechanism, code editor and code completion.

The chosen development environment for this project was Xcode, which was released initially in 2003, as it is the de facto integrated development environment for all Apple software development. It is maintained by Apple and freely available for developers. For a long time, it was the only option developers had, but nowadays some alternatives do exist. Xcode sports a significant number of features such as code editor, code completion, debugger, interface builder, project templates and version control. [4, 3-4.]

2.4 Swift Programming Language

Apple's mobile operating system supports two different programming languages for developing applications: Objective-C and Swift.

Objective-C was developed in the early 1980s by Brad Cox and Tom Love at the Stepstone company (originally called Productivity Products) which they founded in 1983 [5, 166]. A company called NeXT, which was founded by Steve Jobs, licensed a version of Objective-C from Stepstone to build compiler, data management and user interface libraries on top of it. In the late 1990s, Apple acquired NeXT and made Objective-C their main programming language [6, 7.] Objective-C held its status until 2014 when Apple introduced a new programming language called Swift [7, 1].

The Swift programming language is the successor of Objective-C. It was released by Apple at the Apple Worldwide Developer Conference (WWDC) in June 2014. The language has modern syntax and it aims to be a safe, fast and expressive general programming language. Furthermore, Swift is an open source programming language which Apple is developing together with the community [7, 21]. In the Swift Programming Language book, Apple describes Swift as the future of Apple software development.

Swift 3.1 was used to develop the scent training application as it was the latest version available.

2.5 Model-View-Controller Design Pattern

The Mode-View-Controller (originally called Model-View-Editor) pattern was invented by Reenskaug while he was working as a visiting scientist at Xerox in 1978. The purpose of the pattern was to allow its users to manage large and complex data sets. [8.]

Model-View-Controller (MVC) is Apple’s recommended architecture for Cocoa applications. MVC is a prevalent design pattern that adds structure to the application in question. The pattern can be applied to almost any complex application that has a graphical user interface, splitting the application into more manageable decoupled sections. Each of the sections has their own role and well-specified relationship with other sections. Sections interact with each other without knowing the implementation details of the section they are interacting with, thus allowing developers to update a single section without affecting the others. Being able to separate components from each other allows easier testing, maintaining and developing of the application. [9, 527.]. Figure 2 illustrates different components of MVC and their relations.

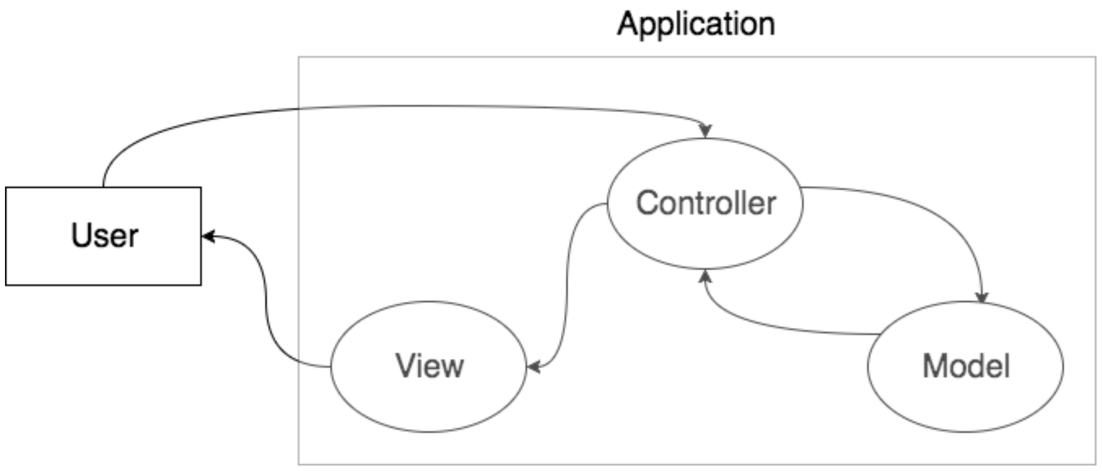


Figure 2. Illustration of the Model-View-Controller design pattern

Looking at figure 2, it can be seen that when a user interacts with the MVC application, the interaction is first received by a controller. The controller, containing the logic to handle user interaction, asks the model to update its state based on what the user did. Once the model has updated its state, the controller gives the updated application a state to a view which updates the user interface.

2.6 Functional Reactive Programming

Functional reactive programming (FRP) had its first appearance in 1997 when Conald Elliott and Paul Hudak released their paper Functional Reactive Animation [10]. As the name states, FRP combines aspects of functional programming and reactive programming.

Functional programming is a paradigm that emphasizes immutable data structures and compositionality and avoids a shared state. Reactive programming is a term used when a program is event based, responds to input and is viewed as a flow of data. Functional reactive programming is a type of reactive programming that enforces the rules of functional programming. [11, 3.] Another definition for functional reactive programming is that it is a declarative programming language allowing application developers to think on a higher abstraction level [12, 1].

Reactive components of the scent training application were developed using a functional reactive programming framework called ReactiveSwift. The framework was chosen due to its popularity and maturity. The framework's GitHub repository had its first commit in March 2012, making it five years old at the time of writing this thesis. Table 1 describes some core components provided by ReactiveSwift.

Table 1. ReactiveSwift's core components.

Component	**Description**
Signal	Observable stream of events over time
Event	Presents occurred matter, for example completion of the signal or next emitted value
SignalProducer	Creates a new signal on observation and performs a side effect the result of which is sent to the signal

To further elaborate the components given in table 1, a practical example of a signal could be the position of a cursor over time. This signal would emit events containing the cursor's coordinates. SignalProducer for example could represent a network request that starts when an observer is added and sends the response of a request to the stream once the network call is complete.

In addition to the above mentioned core components, ReactiveSwift provides a large quantity of operators that can be used to transform signals. Table 2 lists some of the operators and their descriptions.

Table 2. ReactiveSwift’s core components.

Operator	**Description**
map	Transforms values in an event stream based on a supplied function
filter	Filters events in an event stream based on a supplied predicate
combineLatest	Merges multiple streams into a single stream combining the latest values of inner streams
flatMapError	Catches an error in an event stream and replaces it with a SignalProducer instance

The operators listed in table 2 allow a developer to transform signals in various useful ways. For example, a signal could emit product identifiers, which then would be transformed into objects using a `map` operator, then filtered based on the object's properties utilising the `filter` operator and finally combined together with associated information from another signal using the `combineLatest` operator. If the signal has potential to emit an error, it could be caught and handled with the `flatMapError` operator.

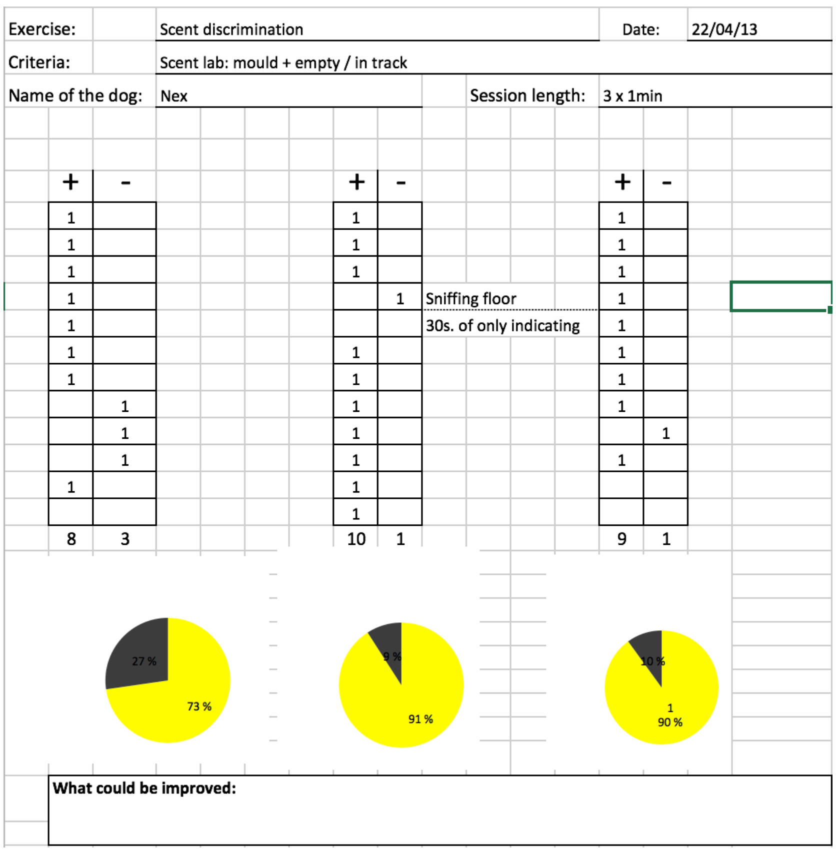
2.7 Reinforcement Table

Wise Nose is Finland’s Smell Detection Association that organises a variety of dog olfactory discrimination related trainings and is the organisation responsible for providing dogs with basic scent discrimination training for the Helsinki University’s research group.

Wise Nose’s training method combined with training methods presented in many research papers was used as a starting point for the scent discrimination training application. Wise Nose’s scent discrimination training includes four different phases with each phase consisting of different training types. The first stage is learning to touch, the second stage is learning how to indicate the target scent, the third stage is learning to detect smell and the last stage is learning to perform scent discrimination.

A reinforcement table is used by dog trainers to follow the training progress of a dog. The reinforcement table, illustrated in figure 3, was used as a starting point to design the scent training application developer during this final year project. In figure 3, three training sessions are described each lasting one minute, aiming to train scent discrimination by using a container with mould and empty control containers. In addition, the figure shows that the dog's name was Nex and that the training was conducted on 15 April 2013.

Each of the training sessions consisted of up to 12 rounds practicing with a similar training setup. Each round could be marked as success or fail if the dog did not find the target scent. In case of failure, the reason was noted. In the first session, the dog had to be able to recognise the target smell from a lineup of five slots, of which one contained the target scent and others were left empty. In the second and third session, one of the slots contained the target scent and the rest of the slots had empty containers.



Exercise:	Scent discrimination	Date:	22/04/13
Criteria:	Scent lab: mould + empty / in track		
Name of the dog:	Nex	Session length:	3 x 1min

+	-	+	-		+	-
1		1			1	
1		1			1	
1		1			1	
1			1	Sniffing floor	1	
1				30s. of only indicating	1	
1		1			1	
1		1			1	
	1	1			1	
	1	1				1
	1	1			1	
1		1				
		1				
8	3	10	1		9	1

27 %
73 %
9 %
91 %
10 %
1
90 %

What could be improved:

Figure 3. Reinforcement table used by dog trainers to follow the training progress of the dog

Looking at the reinforcement table, researchers can easily see the success rate of the training and based on it whether to move onto the next level or to keep practicing the current one.

2.8 Review of Related Studies

This chapter will review two studies describing how dogs were trained to detect the smell of cancer. The review will concentrate on the used training methods, the setup of the scent lineup and the content of training phases.

McCulloch et al. trained five dogs to detect the scent of lung and breast cancer using operant conditioning. Scent training was divided into three training phases all using the same scent lineup. The training phases were considered complete when the dog successfully identified the target scent in at least 30 consecutive trials. The scent lineup incorporated five scent containers of which a single container held the target scent. [13.]

During the first training phase, the dog was taught how to indicate the target odour. The dog was introduced to the training scent lineup that comprised five scent containers: four empty scent containers and one container with a cancer sample and a piece of food to arouse the dog's interest. When the dog sniffed a container holding the target smell, a clicker was used to give an acoustic signal indicating correct behaviour. Immediately after the acoustic signal the trainer commanded the dog to sit. A praise and food reward was given when the dog sat down. [13.]

On the second phase of the training, the trainer did not give any verbal commands to the dog. When the dog correctly indicated the container holding the cancer sample and a piece of food, an acoustic signal was given and the dog was rewarded similarly to the previous training phase. The third training phase was similar to the second phase except that the target container held only the cancer sample without food. [13.]

The second reviewed study is authored by Walczak et al. In the research they trained six dogs to detect the smell of breast cancer, melanoma and lung cancer from breath samples using operant conditioning. Operant conditioning was accomplished using a clicker and food rewards. The training lineup consisted of five scent containers. The scent training comprised of initial training and three training phases. Each training phase was considered complete when the dog performed 40 correct indications in at least 100 trials. [14.]

In the initial training phase, the dog was taught how to indicate the scent container holding the target odour, which in the initial training phase was dog food. The dog was encouraged to sniff all the containers and to indicate the target container holding the dog food by sitting in front of it. On successful indication, the clicker was used to confirm the correct behaviour and the dog was rewarded with food. [14.]

In the first training phase, a single scent container held cancer odour and food, the remaining containers were left empty. The trainer knew the location of the target container, and when the dog sniffed the correct container, the trainer issued a 'sit' command. After the dog sat down, the trainer issued an acoustic signal using a clicker and the dog got rewarded with food and a praise. On each round, the location of the target scent was chosen randomly. The second phase of the training was otherwise the same as the first phase except that the target scent was no more coupled with food scent. In the last phase of the training, the target scent was placed among four control scents. [14.]

Table 3 summarises the characteristics of the trainings seen in the reviewed studies.

Table 3. Summary of reviewed studies

	McCulloch et al. [13]	**Walczak et al. [14]**
Training method	Operant conditioning	Operant conditioning
Number of stations in the training lineup	Five	Five
Number of training phases	Three	Initial + three

Based on table 3, it can be concluded that McCulloch et al. [13] and Walczak et al. [14] have successfully used similar training methods teaching dogs scent detection. In addition to Wise Nose’s training methods, the reviewed studies were used to design the training structure for the scent discrimination training application developed in this final year project.

3 Methods and Features

3.1 Software Specification

3.1.1 Training Process Overview

Training scent discrimination requires a scent lineup composed of a varying number of scent containers. Preferably the containers are made out of machine washable material allowing removal of scent traces that could hinder the training progress. Depending on the learning objective and performed exercise, the containers are empty, hold target scent or control scent. The number of containers in the lineup is increased as exercises get more demanding.

The training process of the developed scent training application was designed to compose of three high-level learning objectives. Each of the learning objectives consists of steps which are composed of multiple exercises. Figure 4 illustrates the structure of the designed learning objectives.

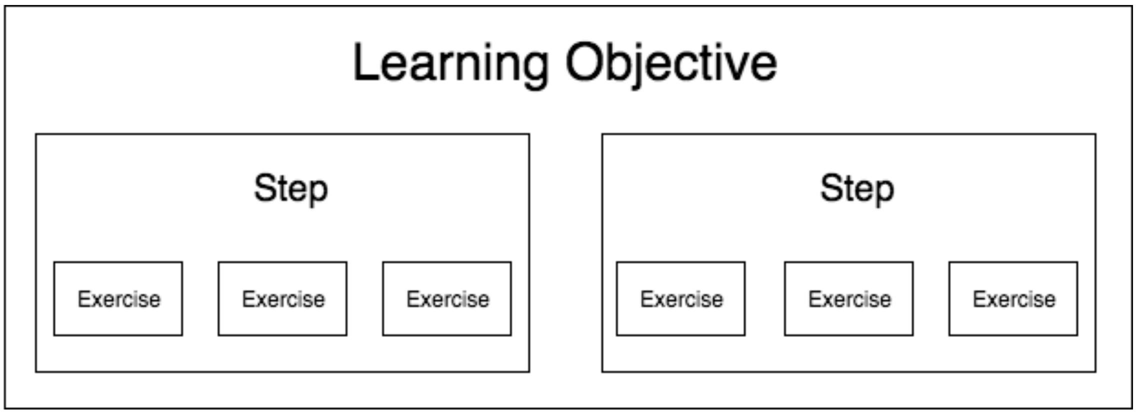


Figure 4. Illustration of learning objectives

The training will begin by choosing a learning objective, then performing exercises in the first step. Once the exercises have been completed, the training will proceed to the next step. Once all steps have been completed, the learning objective is regarded completed and training will continue to the next learning objective.

The scent discrimination has the following learning objectives: learning to indicate, learning the target scent and finally learning to discriminate the target scent from controls. The training objectives are illustrated in figure 5.

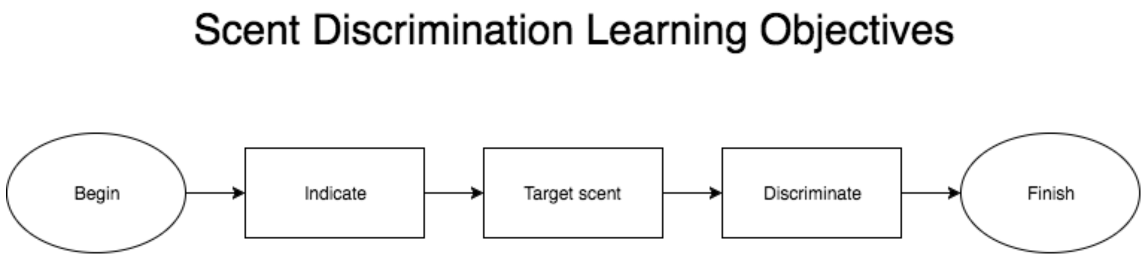


Figure 5. Illustration of training process

As seen in figure 5, the first learning objective is to learn how to indicate the target scent. When it comes to this learning objective, the dog is taught how to behave when it finds the target scent. The indication method is not defined by the application, thus allowing the dog trainer to choose a suitable indication method for the dog. Some of the possible indication methods include touching the scent container with a paw, laying down in front of the container and holding the nose above the container for a given period of time.

During the second learning objective, the dog will familiarise itself with the target scent by practicing on a lineup consisting of a few empty containers and a single container holding the target scent. Training will proceed to the next learning objective once the dog has successfully learned the desired target scent.

The last and most demanding learning objective is scent discrimination. During the learning objective, the dog will learn to differentiate between control scents and the target scent. The scent discrimination training can be considered completed after all the exercises in terms of this learning objective are completed.

3.1.2 Scent Discrimination Training Task Flow

The training task flow starts by a user opening the application. The user then proceeds to choose the dog they would like to train, the current learning objective and a step. After the trainer has chosen the step, she continues to one of the exercises. After the exercise is done, the trainer writes down her improvement suggestions and views the overall training progress of the dog. The flow of the scent discrimination training task is illustrated in figure 6.

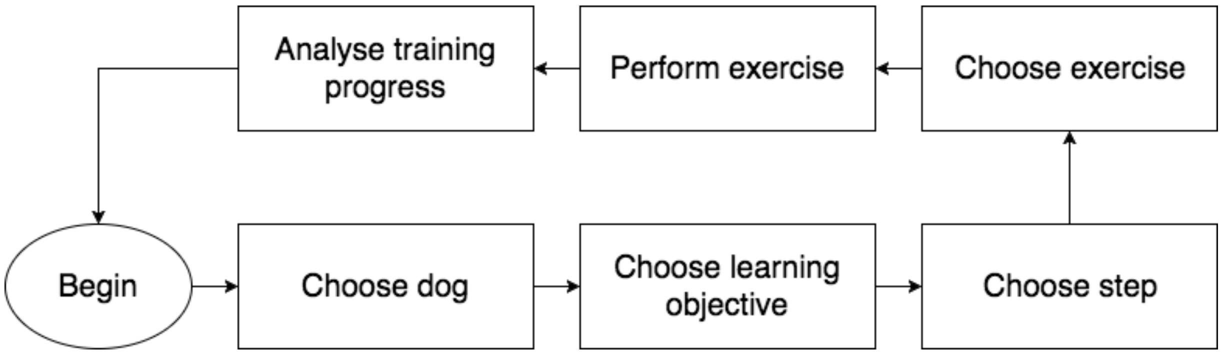


Figure 6. Illustration of a typical application flow

For researchers, the training flow contains additional steps. As figure 6 demonstrates, after choosing an exercise, the researchers can browse sample selection and choose target and control samples placed in the training lineup. After performing the exercise, the application allows researchers to mark which samples the dog indicated to contain target scent.

3.2 Implementation of the Specification

Due to time limitations, the scent training application was initially developed so that it could be later enhanced to meet research purposes. It was decided that the first features to be implemented were those of hobbyists as they comprised a strict subset of features required by researchers and would produce a minimum viable product.

3.2.1 Selecting a Dog

When a user starts the application, the first screen shown is a home view which allows the user to select a dog. If the dog does not yet exist in the system, the user can add it by tapping the “add dog” button in the top right corner. The home screen is seen in figure 7.

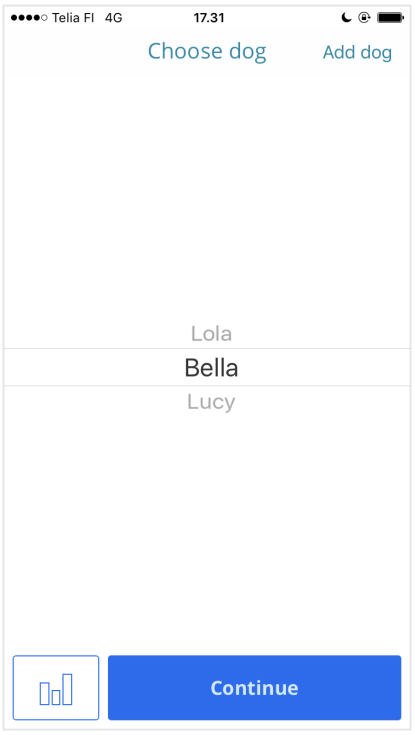
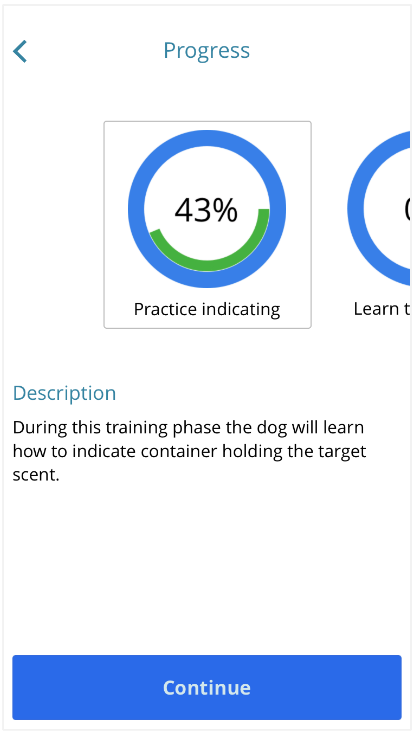


Figure 7. The home screen of the scent discrimination application

After selecting the dog, the user can continue to the training section of the application by pressing continue or browse earlier training results by tapping the analytics button seen in the left bottom corner in figure 7.

3.2.2 Selecting a Training Phase

Once the user has selected a dog and continued to the training, the application will display all training phases to the user. Each of the shown training phases has a percentage label indicating the selected dog's progress on that particular training phase. The view is seen in figure 8.



Description

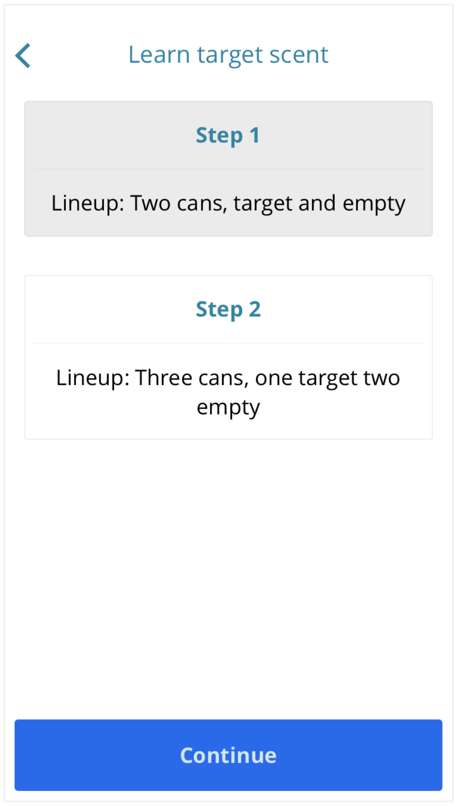
During this training phase the dog will learn how to indicate container holding the target scent.

Figure 8. Training phases view

Once the user selects a training phase, the description for the selected training phase is shown, as seen in figure 8. The description contains information on the purpose and goal of the training phase. The user continues to the next part by tapping the continue button.

3.2.3 Selecting a Training Step

After a training phase has been selected and the user has decided to continue, the application will display each training step belonging to the chosen training phase. The training steps view is seen in figure 9.


Learn target scent

Step 1

Lineup: Two cans, target and empty

Step 2

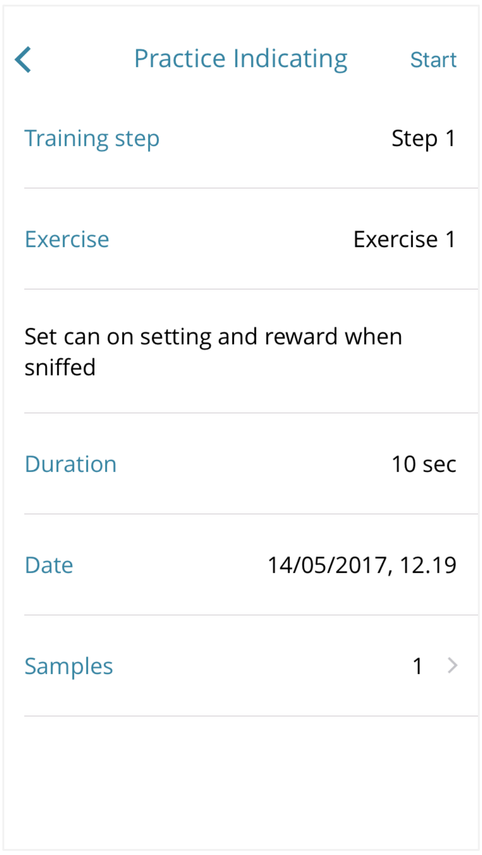
Lineup: Three cans, one target two empty

Figure 9. Learning steps

Each of the training steps contains a description of how the scent training lineup should be set up. For example, as seen in figure 9, learning the first training step of the target scent training requires a scent lineup consisting of two scent containers: a container holding target scent and another empty container.

3.2.4 Starting a Training Session

Once the training phase and training step have been chosen, the training application will present a table view containing options for exercise: training duration, date and optional samples used in the training. The view is illustrated in figure 10.



Practice Indicating Start

Training step	Step 1
Exercise	Exercise 1
Set can on setting and reward when sniffed	
Duration	10 sec
Date	14/05/2017, 12.19
Samples	1

Figure 10. Training session settings

From all options seen in figure 10, the most important settings for hobbyists are the performed exercise and its duration. Description of the exercise is shown below the exercise row. In addition, researchers can select which samples are used in the scent training lineup. After all options are chosen, the training session can be started.

3.2.5 Performing an Exercise

When the training session is started, the application presents the user a virtual clicker. When the green area of the clicker view is tapped, the application will give an acoustic signal indicating to the dog that its behaviour was correct. The red area of the virtual clicker is pressed if the dog makes a mistake during the exercise. The virtual clicker view is illustrated in figure 11.

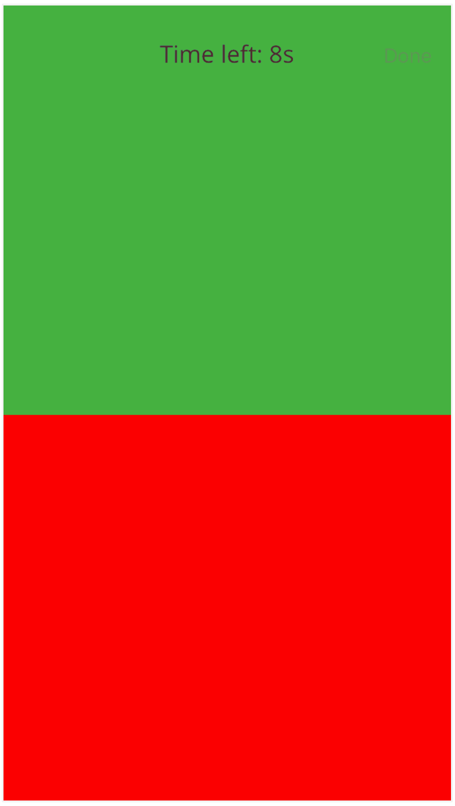


Figure 11. Virtual clicker view

As seen in figure 11, the top of the view contains a label indicating how much time is left for the exercise. After the exercise is completed, the virtual clicker disappears revealing training results and enabling the done button. The result view is illustrated in figure 12.

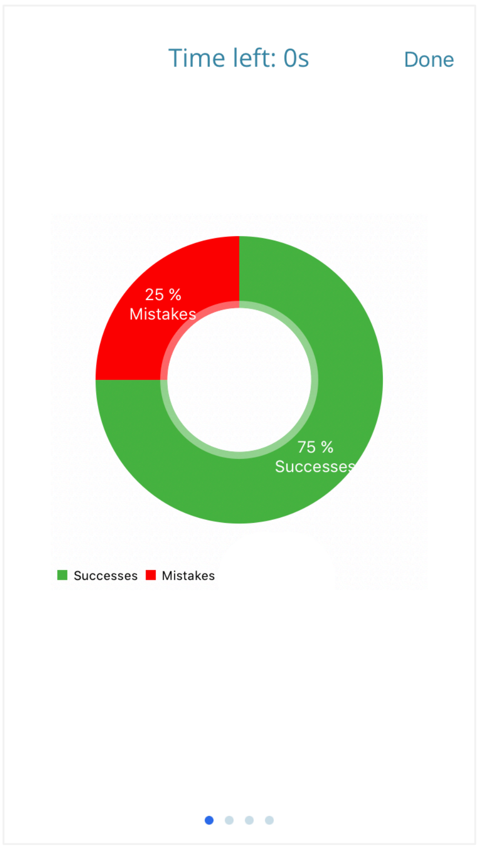


Figure 12. Exercise results view

The exercise result view seen in figure 12 allows the user to perform the exercise again by swiping to the right. The training application allows the user to perform the exercise again up to four times. After the exercise has been conducted for a desired number of times, the training session is finished by pressing the done button in the top right corner. Pressing the done button will take the user to the improvements view.

3.2.6 Recording Improvement Suggestions

After the training session has been completed, the application displays the improvement suggestions view. The improvement suggestions view allows the user to record ideas of how performing the exercise could be improved in the future. For example, different types of rewards could be used for the dog. The improvements view is seen in figure 13.

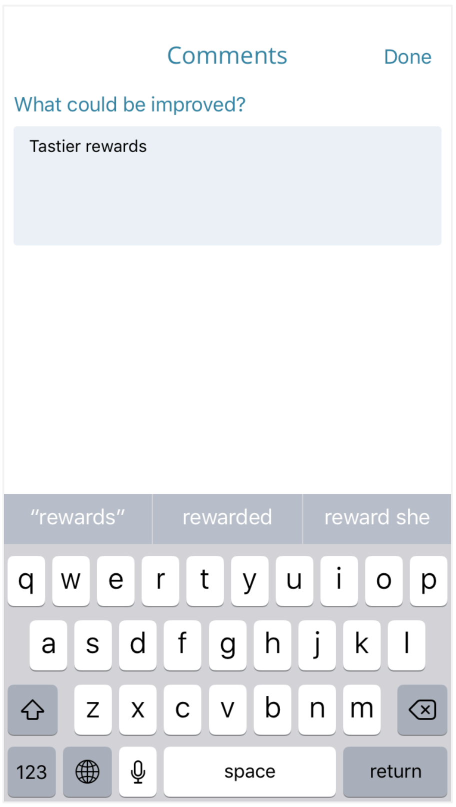
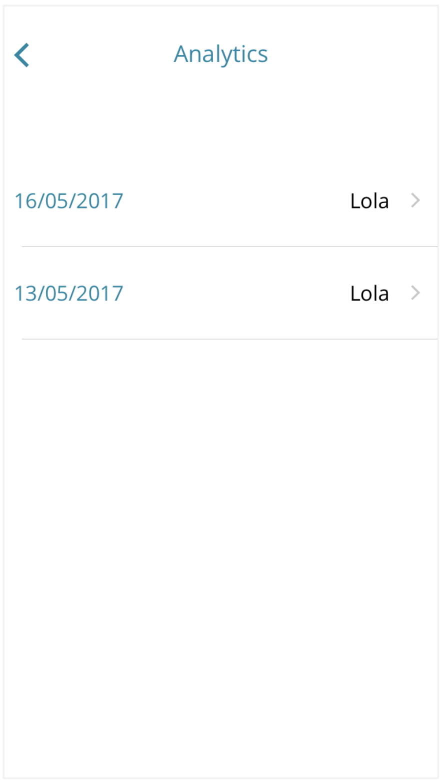


Figure 13. Improvement suggestions view

After the improvements have been entered, the application will take the user back to the home view. From the home view, the user can to continue to the analytics view or perform another exercise.

3.2.7 Browsing Earlier Training Sessions

The training application allows the user to browse earlier performed training sessions. After selecting a dog and pressing the analytics button in the home view, the application will present the user all training sessions performed by the dog. The training session history listing is illustrated in figure 14.



16/05/2017 Lola

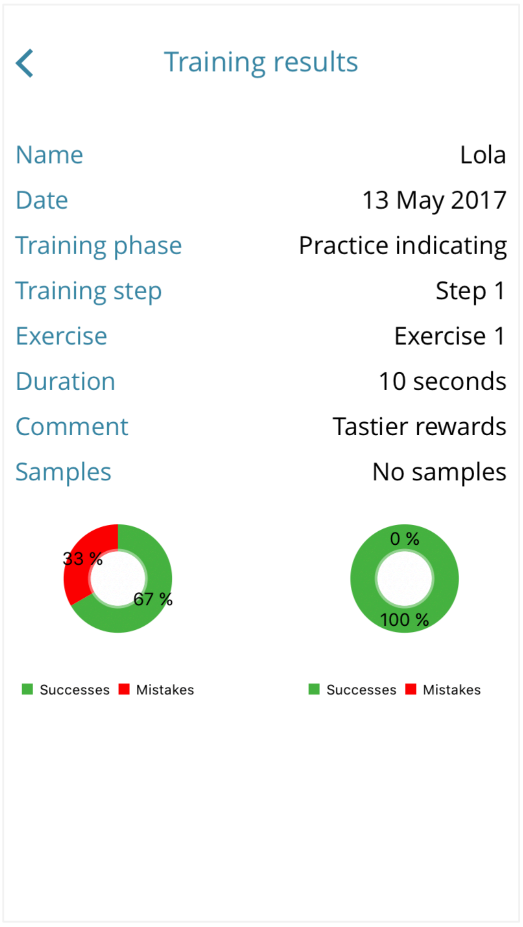
13/05/2017 Lola

Figure 14. Training session history listing view

In the training session history listing view, the performed training sessions are presented in a list form. Each row consists of the name of the dog and a date when the training session was conducted. Training session details can be seen by tapping the corresponding row.

3.2.8 Viewing Training Session Details

The training session details view displays all collected data of a given training session. From the details view, the user can see what the improvement suggestions were, how well the dog performed, what the selected training step was and so on. The training session details view is illustrated in figure 15.


Training results

Name	Lola
Date	13 May 2017
Training phase	Practice indicating
Training step	Step 1
Exercise	Exercise 1
Duration	10 seconds
Comment	Tastier rewards
Samples	No samples

Figure 15. Training session details view

The training session details view is especially useful for researchers as they can see which samples were used during the training.

3.2.9 Browsing Samples

For research purposes the training application allows selecting samples that are used in the training. Figure 16 illustrates the sample browse view. Each sample in the system has a unique identifier that can be used when searching for a particular sample.

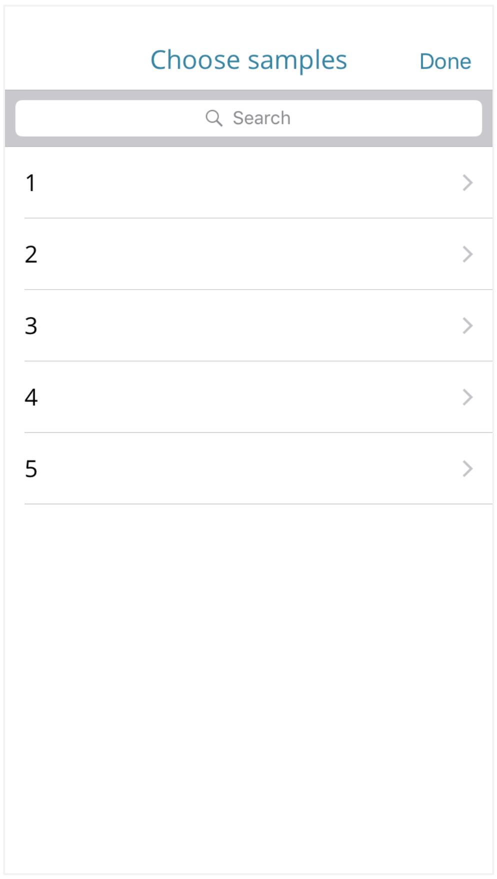
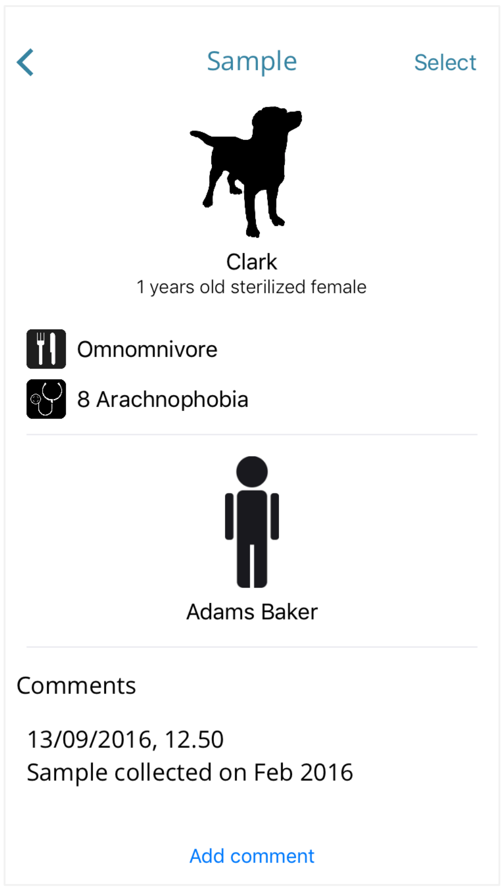


Figure 16. Sample browse view

As figure 16 shows, the application currently contains five samples having identifiers from one to five, shown on the left side of each row. The sample can be selected to be used in the training by going to the sample details view by tapping the row. The sample details view is illustrated in figure 17.



Clark
1 years old sterilized female

Omnomnivore
8 Arachnophobia

Adams Baker

Comments

13/09/2016, 12.50
Sample collected on Feb 2016

Figure 17. Sample details view

In addition to selecting a sample to be used in the scent training, the application allows the user to view sample details and to read and write comments. Sample details provide useful details for research purposes such as sample diagnosis. Additional details to the sample can be written as a comment.

3.3 Software Architecture

3.3.1 Data Model

The data model of the training application is illustrated in figure 18. As shown in the figure, the data model comprises ten classes all related to the scent training process. The classes function as a simple in-memory data store and do not contain business logic.

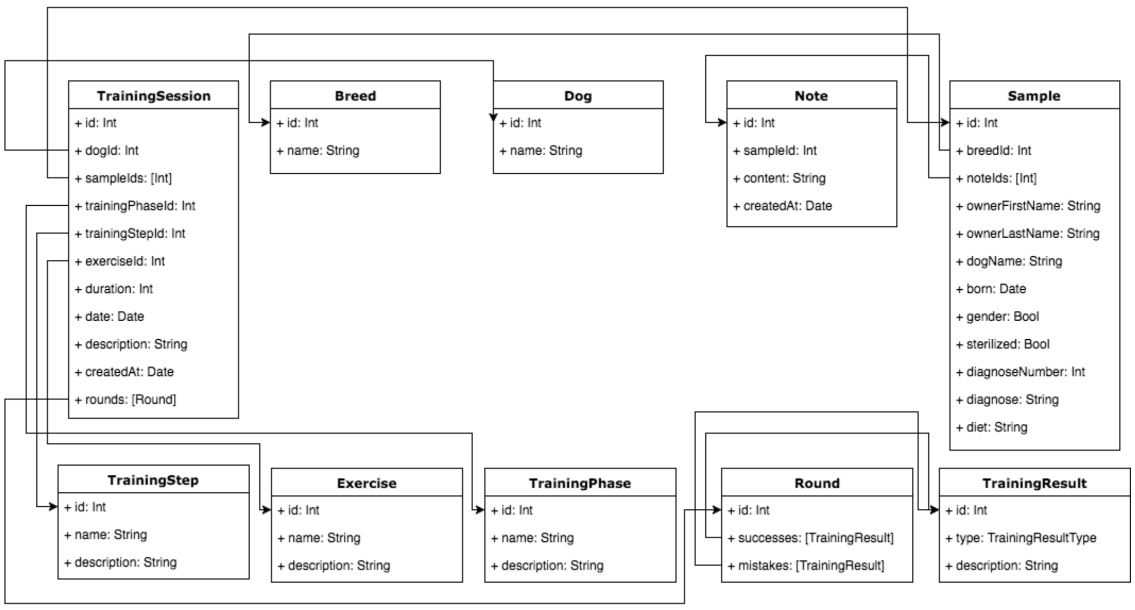


Figure 18. Data model

The number or models and their relations grew organically during the application development as features were added. The most central model is TrainingSession. It represents a single training session thus having a reference to training phase, training step, exercise and any number of performed training rounds.

3.3.2 Service Layer

The responsibility of the service layer is to handle networking with various web application programming interface end points. The service layer of the training application consists of eight classes that all have their own responsibilities. The service layer classes are illustrated in figure 19.

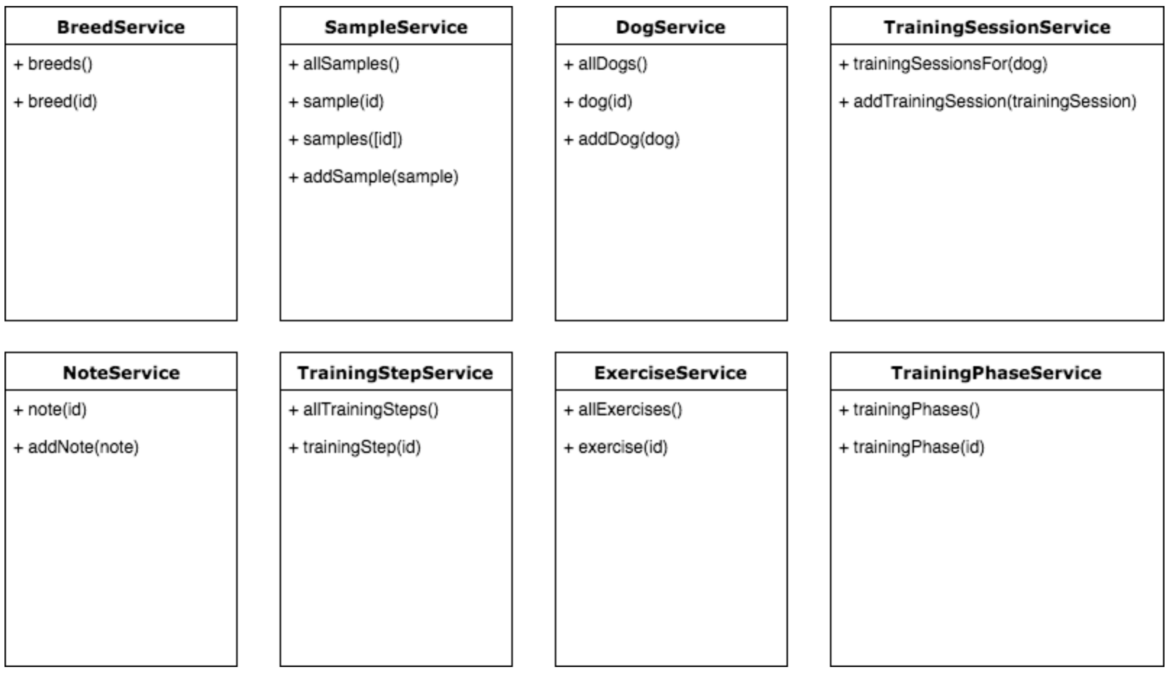


Figure 19. Service layer

The service layer classes utilise a reactive programming paradigm in their internal implementation and expose signals their public interface to simplify asynchronous behaviours. For example, NoteService exposes two methods to its consumers, `note` and `addNote`. The interface of NoteService is illustrated in listing 1. As seen in the listing, both methods return a SignalProducer instance which can be used to subscribe to the events sent by the stream.

```
internal class NoteService {

    static internal let sharedInstance: NoteService.NoteService

    func note(id: Int) -> SignalProducer<Note, NSError>

    func addNote(sampleId: Int, comment: String) -> SignalProducer<Note, NSError>
}
```

Listing 1. NoteService’s public interface

`NoteService.note(id:)` is used by the sample details view to present notes associated with a sample. Before displaying the content of the note to the user, the notes must be first requested from a server based on their ID. This process is illustrated in listing 2.

```
fileprivate func downloadNotes(_ sample: Sample) {
        notes <~ sample
            .noteIds
            .producer
            .take(first: 1)
            .flatten()
            .flatMap(.merge) { return NoteService.sharedInstance.note($0) }
            .on(
                started: { self.commentStackView.displayActivityIndicator(true) },
                completed: { self.commentStackView.displayActivityIndicator(false) }
            )
            .flatMapError { _ in SignalProducer.empty }
            .collect()
}
```

Listing 2. Usage of `NoteService.note(id:)` method

In listing 2, the `downloadNotes(_ sample:)` method first takes a stream emitting an array of note IDs from the given sample model. The signal is then transformed to a new signal, which completes after the first value event is emitted, using the `take(first: 1)` operator. After the transformation, the signal is still emitting an array of IDs but the `NoteService.note(id:)` method needs them one by one so the emitted values are flattened using the `flatten()` operator. Now the signal is emitting one note ID at a time. Each of the emitted note IDs is converted to a SignalProducer using the `flat-map(.merge)` operator. The created SignalProducers are automatically subscribed to, thus starting a network request to download note information from the server for each note ID. A new signal generated by `flatMap(.merge)` will emit note models as soon as they are received from the network.

Side effects are injected to the signal using the `on(started:, completed:)` operator. The injected side effect updates the user interface to display an activity indicator when the first network request is sent and dismisses the activity indicator once all network requests are complete.

The last two operators of the signal chain, `flatMapError()` and `collect()`, will handle potentially emitted an error and collect all note models into an array, respectively. The resulting signal, emitting an array of note models, is then bound to the `notes` property of the view.

4 Discussion

4.1 Results

During this final year project a prototype of scent discrimination training application was developed for the iOS platform based on the designed software specification. The architecture of the application utilized a traditional MVC design pattern to give overall structure to the application and a reactive programming paradigm on the service layer to handle asynchronous network operations.

4.2 Challenges

During the implementation, several challenges were encountered. A new version of the Swift programming language was released during the project. Updating to the new version of the language was a major undertaking breaking many existing features. The migration was completed in a few days. Furthermore, the training application's specification evolved during the project leading to creation of multiple components that were not included in the final product.

4.3 Future development

The training application could be further improved by adding an onboarding experience. The onboarding process familiarizes a new user to the application by displaying a series of views that convey why the user should use the application and how the application is used.

Moreover, to make the application user-friendlier, the exercises could include more detailed instructions on how to perform the exercises. Improved instructions could also illustrate visually how to setup the training scent lineup.

The current implementation of the training application does not take benefit of Apple’s accessibility features. Implementing accessibility is important for various reasons; it democratizes technology and simultaneously contributes to a better user experience for everyone. iOS provides built-in accessibility features for those users that have disabilities. These accessibility features alleviate challenges related to vision, hearing and motor skills. Dynamic Type and VoiceOver are Apple’s accessibility features targeted to empower users with limited vision. Dynamic Type allows users to increase their font size, and VoiceOver reads what is displayed on the screen. For impaired hearing and motor skills, some of the provided features are Live Listen hearing aid and Switch Control letting users to control an application using one or more physical switches.

In addition to built-in accessibility features provided by the operating system, application developers can contribute to the accessibility by considering the application’s colour scheme and by designing simplified user interfaces. The colour scheme should be designed keeping colour vision deficient users in mind. For example, the error or success state in a form should be conveyed in more ways than just colour. For instance, in addition to an error colour, the application could display a text describing the occurred error. Simplified user interfaces benefit users accessing the application through screen readers, such as Apple’s built-in screen reader VoiceOver.

From the training application point of view, accessibility could be enhanced by enabling Dynamic Type on all labels and improve VoiceOver support by adding screen reader context strings to labels and controls. Moreover, the contrast ratio of the application’s colour scheme should be calculated and colours should be reviewed in case the contrast ratio is too low.

5 Conclusion

The goal of this final year project was to develop a prototype of a mobile scent discrimination training application for dogs. The application was to be used in hobby and research context. Based on the given goal, an iPhone application was successfully developed using the Swift programming language, common programming design patterns and reactive programming paradigm.

The developed prototype of a scent discrimination training application allows users to train scent discrimination dogs using a well-defined training process and provides an ability to follow a dog's progress throughout the training.

To prepare the training application for production use, further development should include improvements to user-friendliness and usability testing conducted on hobbyist and research users.

References

1 Apple Inc. Q2 2017 unaudited summary data [online]. Apple Newsroom; 2017. URL: https://www.apple.com/newsroom/pdfs/Q2FY17DataSummary.pdf. Accessed 1 April 2017.

2 Apple Inc. What's new in iOS 10 [online]. Apple Newsroom; 2016. URL: https://www.apple.com/newsroom/2016/09/whats-new-in-ios-10/. Accessed 13 March 2017.

3 Apple Inc. App Store [online]. Support, Apple Developer Center; 2017. URL: https://developer.apple.com/support/app-store/. Accessed 10 March 2017.

4 Knott M. Beginning Xcode. Apress; 2016.

5 Jones C. The technical and social history of software engineering. Addison-Wesley; 2013.

6 Wentk R. Cocoa. John Wiley & Sons; 2010.

7 Mathias M, Gallagher J. Swift programming: the big nerd ranch guide. Pearson Technology Group; 2015.

8 Reenskaug T. The original MVC Reports; 1978.

9 Freeman A. Pro design patterns in Swift. Apress; 2015.

10 Elliott C, Hudak P. Functional reactive animation. ACM SIGPLAN Notices 1997;32(8):263-273

11 Blackheath S Jones A. Functional reactive programming. Greenwich, Connecticut: Manning Pubns; 2015.

12 Czaplicki E. Elm: Concurrent FRP for functional GUIs. Senior thesis, Harvard University; 2012.

13 McCulloch M. Diagnostic accuracy of canine scent detection in early- and late-stage lung and breast cancers. Integrative Cancer Therapies 2006;5(1):30-39

14 Walczak M. Impact of individual training parameters and manner of taking breath odor samples on the reliability of canines as cancer screeners. Journal of Veterinary Behavior 2012;7(5):283-294