Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

Уфимский государственный нефтяной технический университет

Кафедра вычислительной техники и инженерной кибернетики

КУРСОВАЯ РАБОТА

по дисциплине

Мобильные приложения и программирование устройств

Тамагочи

Выполнил студент группы БПО-18-01 Р.У. Мустафин

Принял ст. преподаватель Е.В. Дружинская

Дата представления работы: \_\_\_\_\_\_\_\_\_\_\_\_

Дата защиты: \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Результат: \_\_\_\_\_\_\_\_\_\_

Уфа, 2020 г.

СОДЕРЖАНИЕ

|  |  |  |  |
| --- | --- | --- | --- |
| Введение | | | 3 |
| 1 | Обзор предметной области | | 5 |
| 2 | Проектирование приложения | | 6 |
|  | 2.1 | Функциональное проектирование | 6 |
|  | 2.2 | Эскизирование экранов приложения | 7 |
|  | 2.3 | Логическое моделирование | 8 |
| 3 | Программная реализация | | 10 |

Заключение    23

Список использованной источников         24

**Введение**

В настоящее же время мобильные телефоны прочно укоренились в нашей повседневной жизни благодаря широкому спектру функций. Смартфон в наши дни это, прежде всего: камера, доступ в Интернет и звонки. Помимо все вышеперечисленного зачастую, смартфон используют, как игровую консоль.

В данном приложении воспроизведен всемирно известный “Тамагочи”.

Данное приложение научит следить за питомцем и даст возможность обдумать сможете ли вы ухаживать за ним.

Игра помогает отдохнуть от работы с полезной стороны. В этом и заключается актуальность такого приложения.

Умение заботиться необходимо каждому человеку. В настоящее время практически каждый человек имеет смартфон, поэтому питомец в виде мобильного приложения является лучшим выбором.

Целью курсовой работы является разработка мобильного приложения для ухода за питомцем. Следует определить основные задачи:

* Анализ существующих систем, обладающих схожим функционалом;
* Определение функционала приложения на основе анализа существующих систем;
* Изучение и выбор средств для осуществления разработки мобильного приложения для ОС Android;
* Проектирование разработки приложения;
* Разработка приложения для ухода за питомцем

**1 Обзор предметной области**

Виртуальные питомцы получили широкое распространение в 1990-х годах. В 1995 году компания PF Magic выпустила свою первую компьютерную игру из серии [Petz](https://ru.wikipedia.org/wiki/Petz" \o "Petz), позволяющую заботиться, кормить и обучать домашних питомцев и играть с ними. Выросшие питомцы могли заводиться детёнышей. Питомцы в Petz не умирали, но могли убежать от владельца, если тот о них плохо заботился.

В 1996 году компанией [Bandai](https://ru.wikipedia.org/wiki/Bandai" \o "Bandai) был создан виртуальный питомец [Тамагочи](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%BC%D0%B0%D0%B3%D0%BE%D1%87%D0%B8) — портативное устройство в форме яйца с жидкокристаллическим дисплеем и кнопками для взаимодействия. Продолжительность жизни и самочувствие питомца зависели от того, насколько своевременно его владелец удовлетворял его потребности. По разным оценкам за первые два года после выхода было продано от 30 до 40 миллионов экземпляров первой модели. Годом позже команда разработчиков Тамагочи создала новый продукт — [Digimon](https://ru.wikipedia.org/wiki/Digimon" \o "Digimon). В отличие от Тамагочи, который был рассчитан на пользователей всех возрастов, целевой аудиторией Digimon являлись дети школьного возраста. Главной особенностью была возможность соединять устройства между собой и устраивать между «дигимонами» бои .

В том же году валлийский программист Адам Пауэлл придумал сайт [Neopets](https://ru.wikipedia.org/wiki/Neopets" \o "Neopets), позволяющий заводить себе питомцев в виртуальном мире. Сайт был создан Пауэллом совместно с Донной Уильямс и представлен широкой публике в 1999 году. Помимо непосредственно симуляции питомцев, Neopets представлял широкие возможности для взаимодействия между пользователями в виде личной коммуникации, публичных форумов, возможности продавать и покупать предметы, а также принимать участие в совместных играх. Сайт быстро набрал популярность и к 2002 году насчитывал 32 миллиона пользователей.В 2005 году [Nintendo](https://ru.wikipedia.org/wiki/Nintendo" \o "Nintendo) выпустила игру [Nintendogs](https://ru.wikipedia.org/wiki/Nintendogs" \o "Nintendogs) для своей портативной консоли [Nintendo DS](https://ru.wikipedia.org/wiki/Nintendo_DS" \o "). Игра использовала реалистичную графику и [сенсорный экран](https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D0%BD%D1%81%D0%BE%D1%80%D0%BD%D1%8B%D0%B9_%D1%8D%D0%BA%D1%80%D0%B0%D0%BD), с помощью которого стало возможным взаимодействовать с виртуальными собаками с помощью прикосновений. Встроенный микрофон позволил давать питомцам команды голосом. Как и в Petz, виртуальные собаки в Nintendogs не умирали, но из-за плохого ухода становились невоспитанными, грязными и теряли выученные умения.

Виртуальные питомцы получили распространение в виде приложений для смартфонов. В 2013 году появилось приложение [Мой Говорящий Том](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B9_%D0%93%D0%BE%D0%B2%D0%BE%D1%80%D1%8F%D1%89%D0%B8%D0%B9_%D0%A2%D0%BE%D0%BC), наиболее значимой инновацией которого стало умение питомца повторять слова с помощью синтезированного голоса.

1. **Проектирование приложения**

## **2.1 Функциональное проектирование**

В данном приложении пользователь имеет возможность ухаживать за питомцем. При первом запуске приложения пользователь вводит начальные параметры: имя.

Далее с последующим входом в приложение пользователь может выбрать: создать нового питомца или открыть предыдущего из памяти телефона.

На главной активности располагается панель элементов управления, панель статуса жизненных показателей питомца, настройка звука.

При загрузке питомца с ним можно взаимодействовать путем нажатие на него. В таком случае у него увеличивается уровень счастья. Так же можно дать питомцу игрушку, что так же повышает статус счастья.

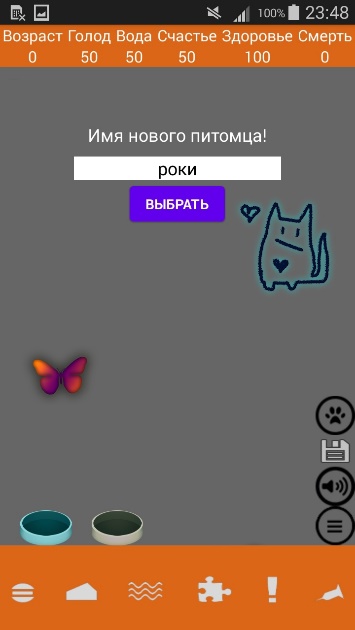
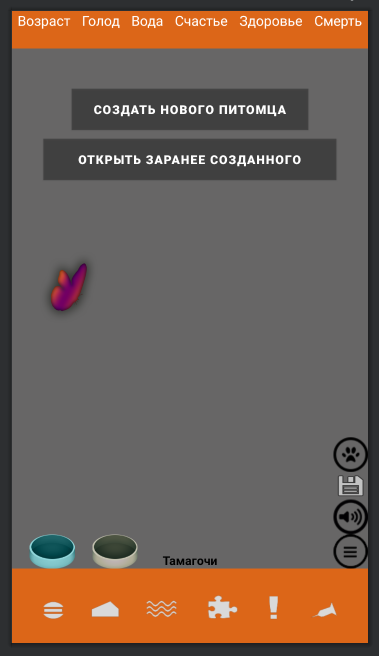
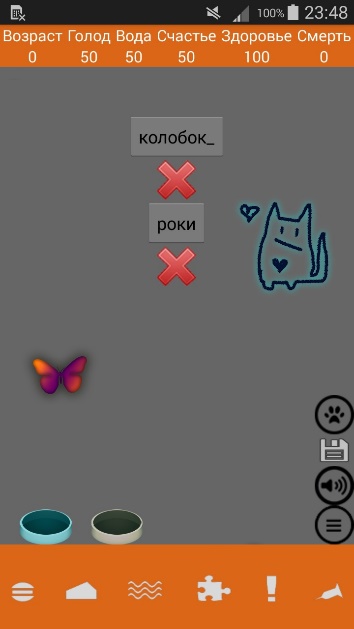
При нажатии на кнопку игры происходит появления двух кнопок. Первая переносит вас в игру угадай число, а вторая в игру угадай слово.

При нажатии на кнопку “сохранения” происходит процесс сохранения прогресса.

Если уровень смерти питомца достигнет значения 100, произойдет окончание игры.

## **Эскизирование экранов приложения**

**Вариант эскиза.** В приложении только 1 активность, так как для отображения полной рабочей области используется включение видимости кнопок. На верхней области показывается жизненные показатели питомца. В нижней области находятся кнопки действий.

****  

## **Логическое моделирование**

1. Класс MainActivity

При запуске приложения открывается класс MainActivity.

На нем будет расположены две кнопки, для первой кнопки привязано действие создание питомца, для второй кнопки предыдущие питомцы которые будут браться из текстового документа.

Основная логика приложения заключена в этом классе. Такие как управление звуком, кнопки взаимодействия с питомцем, кнопка выхода в основное меню и кнопка сохранения

1. Класс Pet.

Имя, здоровье, возраст, счастье, голод, вода, смерть являются объектами класса Pet . В данном классе описываются методы подсчета показателей питомца. К ним мы обращаемся из главной активности, чтобы потом вывести в TextView нужные параметры данных.

1. **Программная реализация**

**3.1 Сохранение параметров**

В приложении требуется сохранить имя, состояние жизненных показателей, чтобы пользователь не начинал с начала. При нажатии на соответствующую кнопку появляется сообщение о сохранении прогресса.



Сохранение параметров:

private void save(){  
 updateText();  
 String petInfo = myPet.getName() + " " + String.*valueOf*(myPet.getAge()) + " " + String.*valueOf*(myPet.getHunger()) + " " +  
 String.*valueOf*(myPet.getThirst()) + " " + String.*valueOf*(myPet.gethealth()) + " " + String.*valueOf*(myPet.getDeath()) + " " +  
 String.*valueOf*(myPet.gethealth()) + " " + String.*valueOf*(myPet.getAgeTime()) + " " +  
 String.*valueOf*((int)(*currentTimeMillis*() / 1000) + " " + myPet.getRandomEvent() + " " + myPet.getCreature());  
 FileOutputStream outputStream;  
 try{  
 outputStream = openFileOutput(myPet.getName() + ".pet", Context.*MODE\_PRIVATE*);  
 outputStream.write(petInfo.getBytes());  
 outputStream.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
}

Чтение значений параметров:

private void choosePet(){  
 File path = getFilesDir();  
 File[] files = path.listFiles();  
 linearLayout.removeAllViewsInLayout();  
 for (int i = 0; i < files.length; i++)  
 {  
 int lengthOfSplit = files[i].getName().split("\\.").length;  
 if(lengthOfSplit > 1 && files[i].getName().toString().split("\\.")[1].equals("pet") && !files[i].getName().toString().split("\\.")[0].equals("null")){  
 name = files[i].getName().toString().split("\\.")[0];  
 final Button newButt = new Button(this);  
 // newButt.setBackgroundResource(R.drawable.petlistimage);  
 newButt.setText(name);  
 newButt.setVisibility(linearLayout.getVisibility());  
 newButt.setLayoutParams(new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*WRAP\_CONTENT*, ViewGroup.LayoutParams.*WRAP\_CONTENT*));  
 final ImageButton deleteButt = new ImageButton(this);  
 deleteButt.setVisibility(linearLayout.getVisibility());  
 deleteButt.setBackgroundResource(R.drawable.*deletebtn*);  
 deleteButt.setLayoutParams(new LinearLayout.LayoutParams(ViewGroup.LayoutParams.*WRAP\_CONTENT*, ViewGroup.LayoutParams.*WRAP\_CONTENT*));  
 newButt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 linearLayout.setVisibility(View.*GONE*);  
 try {  
 view = getCurrentFocus();  
 if (view != null) {  
 InputMethodManager imm = (InputMethodManager)getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
 imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
 }  
 String[] words = new String[11];  
 FileInputStream petFile = openFileInput(newButt.getText().toString() + ".pet");  
 BufferedReader br = new BufferedReader(new InputStreamReader(new BufferedInputStream(petFile)));  
 String line;  
 while ((line = br.readLine()) != null) {  
 words = line.split(" ");  
 }  
 br.close();  
 myPet = new Pet(words[0], Integer.*parseInt*(words[1]), Integer.*parseInt*(words[2]), Integer.*parseInt*(words[3]),  
 Integer.*parseInt*(words[4]), Integer.*parseInt*(words[5]), Integer.*parseInt*(words[6]),  
 Integer.*parseInt*(words[7]), Integer.*parseInt*(words[8]), Integer.*parseInt*(words[9]), words[10]);  
 myPet.updateAwayTime((int)(*currentTimeMillis*() / 1000));  
 resourceID = getResources().getIdentifier(words[10], "drawable", getPackageName());  
  
 creatureImage.setBackgroundResource(resourceID);  
 creatureImage.setVisibility(View.*VISIBLE*);  
 if(myPet.getHappy() <= 80){  
 Toast.*makeText*(getApplicationContext(),"С возращением! счастье (+20)",Toast.*LENGTH\_LONG*).show();}  
 updateGame();  
 save();  
 loaded = true;  
 newButt.setVisibility(View.*GONE*);  
 deleteButt.setVisibility(View.*GONE*);  
 openOldPetButton.setVisibility(View.*GONE*);  
 createNewPetButton.setVisibility(View.*GONE*);  
 linearLayout.setVisibility(View.*GONE*);  
 linearLayout.removeAllViewsInLayout();  
 loadedPetNameText.setText(myPet.getName().replace("\_", " "));  
 loadedPetNameText.setVisibility(View.*VISIBLE*);  
 } catch (Exception e){  
 e.printStackTrace();  
 }  
 }  
 });  
 deleteButt.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 File dir = getFilesDir();  
 try{  
 File file = new File(dir, name + ".pet");  
 if (file.exists()) {  
 file.delete();  
 deleteButt.setVisibility(View.*GONE*);  
 newButt.setVisibility(View.*GONE*);  
 linearLayout.setVisibility(View.*GONE*);  
 linearLayout.removeView(deleteButt);  
 linearLayout.removeView(newButt);  
 linearLayout.removeAllViewsInLayout();  
 choosePet();  
 }  
 } catch (Exception e){  
  
 }  
 }  
 });  
  
 if (linearLayout != null) {  
 linearLayout.addView(newButt);  
 linearLayout.addView(deleteButt);  
 }  
 }  
 }  
}

* 1. **Обновления статуса голода, жажды, здоровья, смерти, счастья**

1. public void updateHunger(int calories, boolean treat){  
    if(hunger + calories > 100) {  
    hunger = 100;  
    }  
    else if(hunger + calories <= 0){  
    hunger = 0;  
    updateHealth(calories);  
    }  
    else{  
    if(treat){  
    hunger += calories;  
    updateHappy(5, false);  
    updateHealth(-2);  
    }  
    else{  
    hunger += calories;  
    }  
    }  
   }  
     
   public void updateThirst(int water){  
    if(thirst + water > 100)  
    thirst = 100;  
    else if(thirst + water <= 0) {  
    thirst = 0;  
    updateHealth(water);  
    }  
    else  
    thirst += water;  
   }  
     
   public void updateHealth(int heal){  
    if(health + heal > 100)  
    health = 100;  
    else if(health + heal <= 0) {  
    updateDeath(heal);  
    health = 0;  
    }  
    else  
    health += heal;  
   }  
   public void updateDeath(int major){  
    major = *abs*(major);  
    if(death + major > 100) {  
    death = 100;  
    }  
    else if(death + major <= 0){  
    death = 0;  
    } else {  
    death += major;  
    }  
   }  
     
   public void updateHappy(int gameCost, boolean isShot){  
    if(happy + gameCost > 100){  
    happy = 100;  
    } else if(happy + gameCost <= 0){  
    if(!isShot)  
    updateHealth(-(*abs*(happy + gameCost)));  
    happy = 0;  
    } else {  
    happy += gameCost;  
    }  
   }
   1. **Обновление возраста**
2. public boolean ageUp(int time){  
    if(age <= 100){  
    if(time - ageTime >= 300){  
    age += (time - ageTime) / 300;  
    ageTime = time;  
    return true;  
    }  
    }  
    return false;  
   }
   1. **События при выполнения действий**

public String randomEvent(){  
 randomEvent = *abs*((int) (Math.*random*() \* events.size()) - 1);  
 if(randomEvent == 0){  
 updateHappy(-10, false);  
 updateHealth(-15);  
 } else if(randomEvent == 1){  
 updateHappy(15, false);  
 } else if(randomEvent == 2){  
 updateHappy(-10, false);  
 }  
 return events.get(randomEvent);  
}

Написаны заранее известные события и с помощью метода randomEvent() они воспроизводятся.

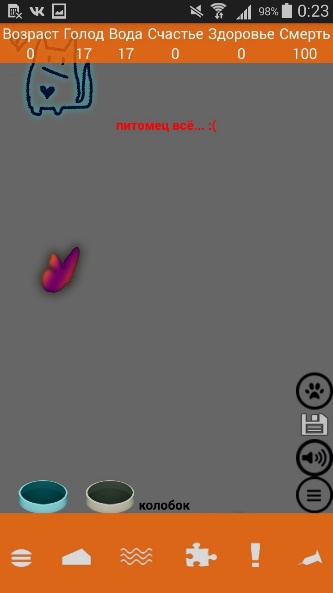
public void onCreate(SQLiteDatabase db) {  
 db.execSQL("CREATE TABLE notes (" +  
 *COLUMN\_ID* + " INTEGER PRIMARY KEY AUTOINCREMENT, " +  
 *COLUMN\_DATE* + " DATE, " +  
 *COLUMN\_WEIGHT* + " TEXT);");  
  
}

В методе **UpdateGame()** реализуется обновление состояние игры

private void updateGame(){  
 myPet.ageUp((int)(*currentTimeMillis*() / 300));  
 handlingDeath();  
 save();  
}

Метод handlingDeath() воспроизводит действие при смерти питомца.

private void handlingDeath(){  
 if(myPet.isDead()){  
 deadPetText.setVisibility(View.*VISIBLE*);  
 openOldPetButton.setVisibility(View.*VISIBLE*);  
 createNewPetButton.setVisibility(View.*VISIBLE*);  
 }  
}



Метод animateCreature() создает передвижение по позиции.

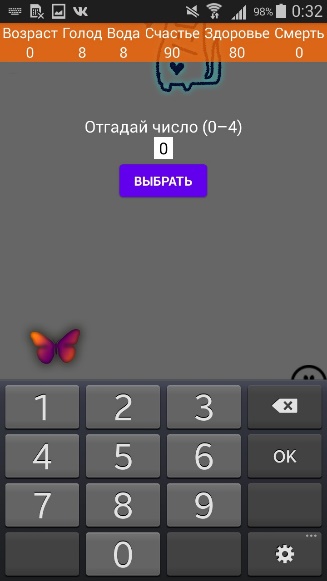
private void animateCreature(){  
 int randomX = 300 + (int) (Math.*random*() \* (300 - (width - 300)) + 1);  
 int randomY = 300 + (int) (Math.*random*() \* (300 - (height - 300)) + 1);  
 creatureAnimator = ObjectAnimator.*ofFloat*(creatureImage, "translationX", randomX);  
 creatureAnimator.setDuration(2000);  
 creatureAnimator.start();  
 creatureAnimator = ObjectAnimator.*ofFloat*(creatureImage, "translationY", randomY);  
 creatureAnimator.setDuration(2000);  
 creatureAnimator.start();  
}

Метод changeSound() управляет звуками в игре. При взаимодействии с питомцем. А так же позволяет включать и выключать звук.

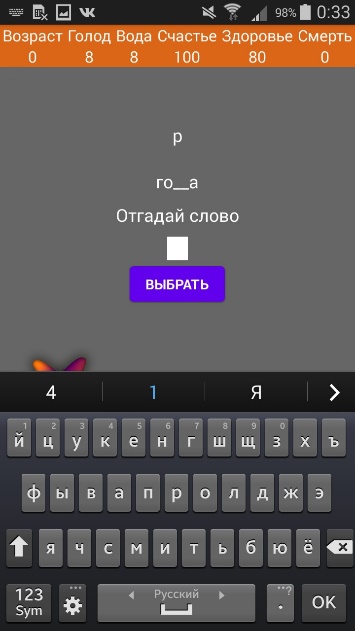
private void changeSound(){  
 if(soundORnot) {  
 catpurrSound.setVolume(1.0f,1.0f);  
 drinkingSound.setVolume(0.4f,0.4f);  
 eatingSound.setVolume(0.4f,0.4f);  
 soundImageButton.setBackgroundResource(R.drawable.*soundbtn*);  
 FileOutputStream outputStream;  
 try{  
 outputStream = openFileOutput("settings", Context.*MODE\_PRIVATE*);  
 outputStream.write("true".getBytes());  
 outputStream.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 soundORnot = false;  
 } else {  
 catpurrSound.setVolume(0.0f,0.0f);  
 drinkingSound.setVolume(0.0f,0.0f);  
 eatingSound.setVolume(0.0f,0.0f);  
 soundImageButton.setBackgroundResource(R.drawable.*mutesoundbtn*);  
 FileOutputStream outputStream;  
 try{  
 outputStream = openFileOutput("settings", Context.*MODE\_PRIVATE*);  
 outputStream.write("false".getBytes());  
 outputStream.close();  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 soundORnot = true;  
 }  
}

Встроенные игры для поднятие счастья питомца реализованы следующим образом. При нажатии первой игры игроку необходимо угадать рандомное значение из диапазона чисел от 0 до 4, во второй игре игроку необходимо угадать буквы из слова.

playImageButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (loaded) {  
 event();  
 gameOneButton.setVisibility(View.*VISIBLE*);  
 gameTwoButton.setVisibility(View.*VISIBLE*);  
  
 gameOneButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
  
 gameAttempts = 3;  
 guessANumberText.setVisibility(View.*VISIBLE*);  
 guessANumberText.setText(R.string.*guess\_a\_number*);  
 numberEditText.setVisibility(View.*VISIBLE*);  
 submitButton.setVisibility(View.*VISIBLE*);  
  
 random = (int) (Math.*random*() \* (0 - 4) + 1);  
 gameOneButton.setVisibility(View.*GONE*);  
 gameTwoButton.setVisibility(View.*GONE*);  
 submitButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(!numberEditText.getText().toString().isEmpty()) {  
 int guessInt = Integer.*parseInt*(numberEditText.getText().toString());  
 if (guessInt == random) {  
 myPet.play();  
 gameAttempts = 0;  
 correctToast.show();  
 numberEditText.getText().clear();  
 } else {  
 gameAttempts--;  
 numberEditText.getText().clear();  
 nopeToast.show();  
 }  
 if (gameAttempts == 0 || gameAttempts == 3) {  
 view = getCurrentFocus();  
 if (view != null) {  
 InputMethodManager imm = (InputMethodManager) getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
 imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
 }  
 guessANumberText.setVisibility(View.*GONE*);  
 numberEditText.setVisibility(View.*GONE*);  
 submitButton.setVisibility(View.*GONE*);  
 updateGame();  
 }  
 }  
 }  
 });  
 }  
 });  
 gameTwoButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 alreadyGuessedLettersText.setText("");  
 gameOneButton.setVisibility(View.*GONE*);  
 gameTwoButton.setVisibility(View.*GONE*);  
 // Words for hangman  
 words = new ArrayList<>();  
 words.add("яблоко"); words.add("хлеб"); words.add("кошка"); words.add("собака"); words.add("слон"); words.add("лошадка");  
 words.add("воздух"); words.add("история"); words.add("лед"); words.add("прыжок"); words.add("сорость"); words.add("мальчик");  
 words.add("месяц"); words.add("осьминог"); words.add("картина"); words.add("звук"); words.add("гонка");  
 words.add("кнопка"); words.add("свет"); words.add("облако"); words.add("солнце"); words.add("дождь"); words.add("кошелек");  
  
 // Words that were obfuscated  
 lettersHidden = new ArrayList<>();  
  
 // Letters Guessed  
 guessedLettersArr = new ArrayList<>();  
  
 // Picking a random word from words  
 random = *abs*((int) (Math.*random*() \* words.size()) - 1);  
 word = words.get(random);  
  
 // Finding letters to obfuscate  
 for (int i = 0; i < 3; i++) {  
 int rando = *abs*((int) (Math.*random*() \* word.length()) - 1);  
 Character randomLetter = word.charAt(rando);  
 if (!lettersHidden.contains(randomLetter)) {  
 lettersHidden.add(randomLetter);  
 }  
 }  
  
 submitButton.setVisibility(View.*VISIBLE*);  
 letterEditText.setVisibility(View.*VISIBLE*);  
 guessALetterText.setVisibility(View.*VISIBLE*);  
 guessALetterText.setText(getString(R.string.*guess\_a\_letter*));  
  
 // Creating an obfuscated word... obfuscate.  
 hiddenWordString = wordChange(word, lettersHidden);  
  
 alreadyGuessedLettersText.setVisibility(View.*VISIBLE*);  
 hiddenWordText.setVisibility(View.*VISIBLE*);  
 hiddenWordText.setText(hiddenWordString);  
 gameAttempts = 6;  
  
 submitButton.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (letterEditText.getText().length() > 0)  
 characterGuess = letterEditText.getText().charAt(0);  
 else  
 characterGuess = ' ';  
  
 letterEditText.getText().clear();  
 if (lettersHidden.contains(characterGuess)) {  
 lettersHidden.remove(characterGuess);  
 hiddenWordString = wordChange(word, lettersHidden);  
 hiddenWordText.setText(hiddenWordString);  
 correctToast.show();  
 } else {  
 guessedLettersArr.add(characterGuess);  
 gameAttempts--;  
 nopeToast.show();  
 }  
  
 alreadyGuessedLettersText.setText(wrongGuesses(guessedLettersArr));  
  
 if (lettersHidden.isEmpty() || lettersHidden.size() == 0) {  
 view = getCurrentFocus();  
 if (view != null) {  
 InputMethodManager imm = (InputMethodManager) getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
 imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
 }  
 lettersHidden.removeAll(lettersHidden);  
 guessedLettersArr.removeAll(guessedLettersArr);  
 myPet.play();  
 correctToast.show();  
 alreadyGuessedLettersText.setVisibility(View.*GONE*);  
 guessALetterText.setVisibility(View.*GONE*);  
 letterEditText.setVisibility(View.*GONE*);  
 submitButton.setVisibility(View.*GONE*);  
 hiddenWordText.setVisibility(View.*GONE*);  
 updateGame();  
 guessedLettersArr.clear();  
 } else if (gameAttempts == 0) {  
 view = getCurrentFocus();  
 if (view != null) {  
 InputMethodManager imm = (InputMethodManager) getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
 imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
 }  
 alreadyGuessedLettersText.setVisibility(View.*GONE*);  
 lettersHidden.removeAll(lettersHidden);  
 guessedLettersArr.removeAll(guessedLettersArr);  
 nopeToast.show();  
 guessALetterText.setVisibility(View.*GONE*);  
 letterEditText.setVisibility(View.*GONE*);  
 submitButton.setVisibility(View.*GONE*);  
 hiddenWordText.setVisibility(View.*GONE*);  
 updateGame();  
 guessedLettersArr.clear();  
 }  
 }  
 });  
 }  
 });  
 updateGame();  
 animateCreature();  
 }  
 }  
});



Игра 1.



Игра 2.

ЗАКЛЮЧЕНИЕ

Результатом работы является мобильное приложение, обладающее функционалом, решающим поставленную цель.

Данное приложение повышает уровень ответственности человека. Может заменить домашнего питомца, если нет возможности такового завести. Данное приложение я рекомендую для людей которые хотят завести домашнего питомца.

С точки зрения дизайна, изменилось многое, в сравнении с намеченными эскизами. Был добавлен новый фон, заменены кнопки и сам питомец.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Амелин К.С. Введение в разработку приложений для мобильных платформ: К.С. Амелин, О.Н. Граничин, В И. Кияев, А.В. Корявкою.– СПб: Издательство ВВМ, 2011.— 507 с
2. MedUniver [Электронный ресурс]. – URL: <https://meduniver.com/Medical/profilaktika/indeks_massi_tela.html> (дата обращения 22.05.2021)
3. Освой программирование играючи [Электронный ресурс]. – URL: <http://developer.alexanderklimov.ru/> (дата обращения 22.05.2021)
4. [METANIT.COM](https://metanit.com/) Сайт о программировании [Электронный ресурс]. – URL: <https://metanit.com/> (дата обращения 22.05.2021)
5. Тысячи бесплатных иконок [Электронный ресурс]. – URL: <https://icon-icons.com/ru/> (дата обращения 22.05.2021)