Пловдивски Университет „Паисий Хилендарски”
Факултет по Математика, Информатика и Информационни технологии

Дипломна работа

Тема: Мобилна тестова система

Дипломант:
Александър Руменов Инджов
Фак. № 1301681081

Научен ръководител:
ас. Йордан Тодоров

С Ъ Д Ъ Р Ж А Н И Е

1 Увод

През последните години, все повече се повишава вниманието към технологизацията на образованието. Това може да се обясни с нарастването на неговата роля за социално-икономическото развитие на обществото. Все по-определено се осъзнава, че от равнището и качеството на образованието зависи състоянието на цялото общество. Развитието на образованието способства за развитието на науката, което на свой ред осигурява разработка и внедряване на нови технологии.

С течение на времето метода за изпитване, се е променил, от обикновенното „изпитване на дъската", той е прераснал в електронен вариант, декстоп приложения и дори мобилни приложения които преспокойно и доста удобно се използват.

Докато в днешно време използването на мобилни телефони или лаптопи в училище най-често е забранено, скоро може да заменят учебниците, тетрадките и дори учителя.

Едно от основните направления в развитието на образованието е към интегриране на популярни и масово използвани електронни технологии, каквито подрастващите обичат в учебните курсове. Освен това те могат умело и без проблем да се интегрират и в Предучилищно обучение, Началното образование , Висшето образование , Обучителни организации и професионално образование и дори те могат да се използват за обучения на учителите.

Всяка човешка дейност, независимо от какъв вид е тя, има своя технологическа страна, извършва се в една или друга форма, протича по един или друг начин, има свои етапи, провежда се в съответствие с едни или други методи и средства и освен обучение технологиите позволяват и много по-различен начин на оценяване на учениците. Има апликации, в които учителите могат да видят в рамките на минути кои деца на кой тип въпроси се справят по-добре. Има и анализи как се справя целият клас като група, както и за всеки ученик поотделно.

В тази посока на мислене е и приложението, което днес ще се опише.

Това приложение, което е представено тук ще бъде бъдещо клиент-сървър приложение, като сървърната част ще бъде написана на JAVA и тя ще представлява няколко SERVLET-А. Цялостната база данни ще бъде изнесена за ползване от единият от SERVLET-ИТЕ,

а другите SERVLET-И може да изпращат обратно отговори(response) на заявкатаи изпратена от клиента (request) за да се извърши комуникация.
Клиента ще бъде приложение за телефон работещ с Андроид операционна система.

1.1 Цел

Изискванията за софтуера, посочени в този документ са за апликация – Мобилна тестова система специално за Андроид операционна система. Тя е предназначена за използването (по-скоро бъдещото използване) от:

деца от Началното училище
ученици от Средното училище
ученици от Основно училище
студенти от Висшето образование

и е удобно средство за проверка на знанията и научения материал по избран от тях дисциплина.Съответно с лека корекция тази апликация може да се превърне в игра дори и за по – малки деца в Детската градина.

1.2 Хардуерни изисквания

Първоначално за приложението ще е необходимо устройство на което да го инсталирате и използвате. Както се споменава по-долу по-добре е да имате телефон с андроид операционна система, а телефон с по-голяма версия андроид операционна система.
При възможност, че приложението не може да се набави от студента чрез изтегляна на необходимата версия тогава ще му се наложи чрез USB или Bluethoot да се свърже с преносимия си компютър за да го качи на телефона и съответно да го инсталира и използва

Може също да се свали от github с git bash, като се изплзва командата git clone
и за линк се използва:https://github.com/sashe944/apkFile

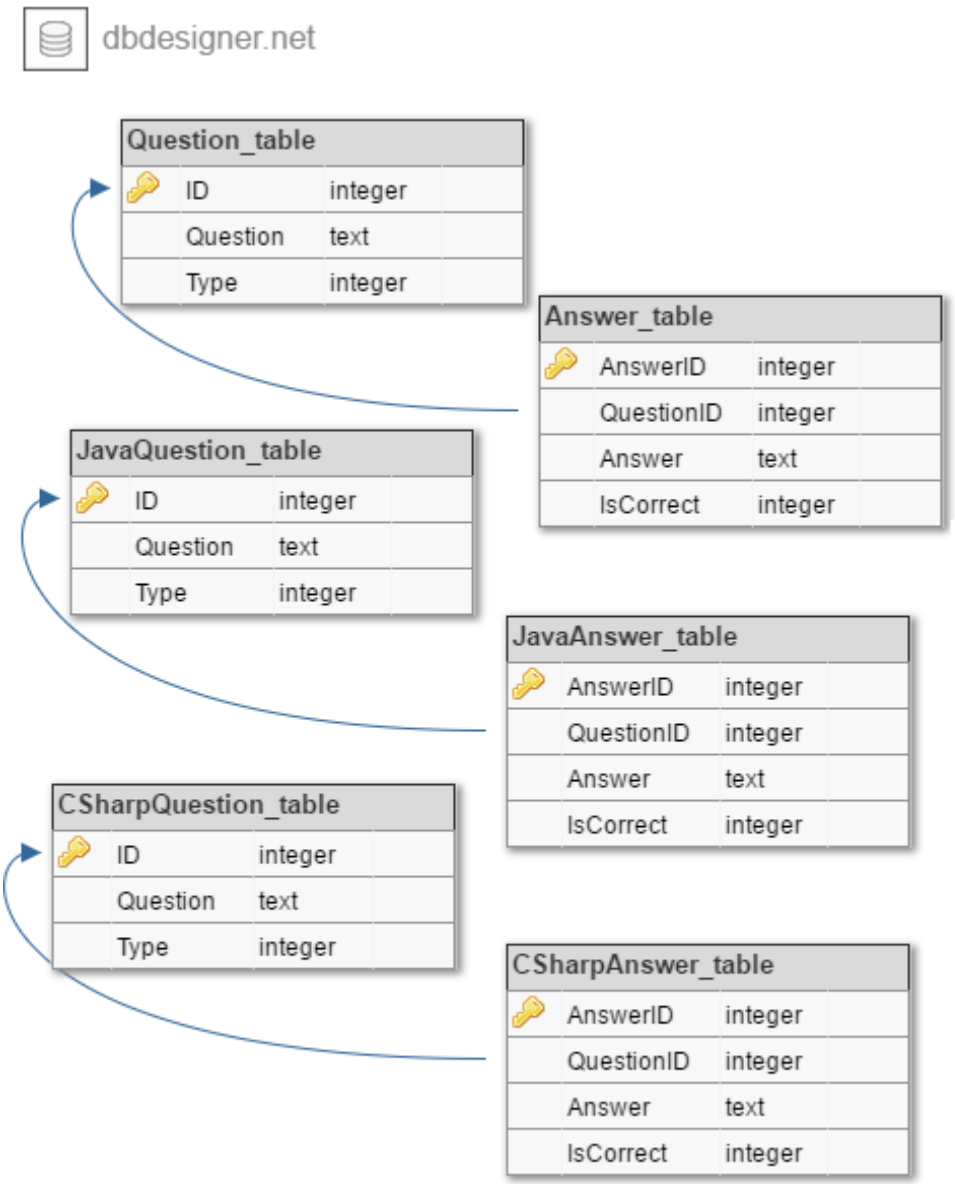
1.1 Софтуерни изисквания

За да се инсталива клиентското приложение студентът трябва да има телефон с андроид операционна система, която „съпортва” поне API LEVEL 19 (така познатат

версия kit kat 4.4), но е за предпочитане и по виско API LEVEL заради добавените нови ефекти, които се потдържат в по – високите LEVEL-И

2 Архитектура

2.1 Диаграма на базата данни за сървъра



2.2 Таблици описваща по-подробно базата данни на сървъра

ме на таблицата	Полета в таблиците	Цел на Полета
Question_table	ID INTEGER PRIMARY KEY AUTOINCREMENT	ID на въпроса за Android които се увеличава с 1 на всеки нов въпрос автоматично
	Question TEXT	Поле за запазване на въпрос за теста
	Type INTEGER	Поле за тип на въпроса
Answer_table	AnswerID INTEGER PRIMARY KEY AUTOINCREMENT	ID на отговора които се увеличава с 1 на всеки нов отговор автоматично
	QuestionID INTEGER	ID на зададения въпрос
	Answer TEXT	Отговор на зададения въпрос
	IsCorrect INTEGER	Поле с което се отбеязва правилния отговор с 1 за правилен и 0 за неправилен
CSharpQuestion_table	ID INTEGER PRIMARY KEY	ID на въпроса за C#

	AUTOINCREMENT Question TEXT Type INTEGER	които се увеличава с 1 на всеки нов въпрос автоматично Поле,което запазва въпрос за теста Поле за тип на въпроса
CSharpAnswer_table	AnswerID INTEGER PRIMARY KEY AUTOINCREMENT QuestionID INTEGER Answer TEXT IsCorrect INTEGER	ID на отговора които се увеличава с 1 на всеки нов отговор автоматично ID на зададения въпрос Отговор на зададения въпрос Поле с което се отбеязва правилния отговор с 1 за правилен и 0 за неправилен
JavaQuestion_table	ID INTEGER PRIMARY KEY AUTOINCREMENT Question TEXT	ID на въпроса за Java които се увеличава с 1 на всеки нов въпрос автоматично Поле за запазване на

		въпрос за теста
	Type INTEGER	Поле за тип на въпроса
JavaAnswer_table	AnswerID INTEGER PRIMARY KEY AUTOINCREMENT	ID на отговора които се увеличава с 1 на всеки нов отговор автоматично
	QuestionID INTEGER	ID на зададения въпрос
	Answer TEXT	Отговор на зададения въпрос
	IsCorrect INTEGER	Поле с което се отбеязва правилния отговор с 1 за правилен и 0 за неправилен

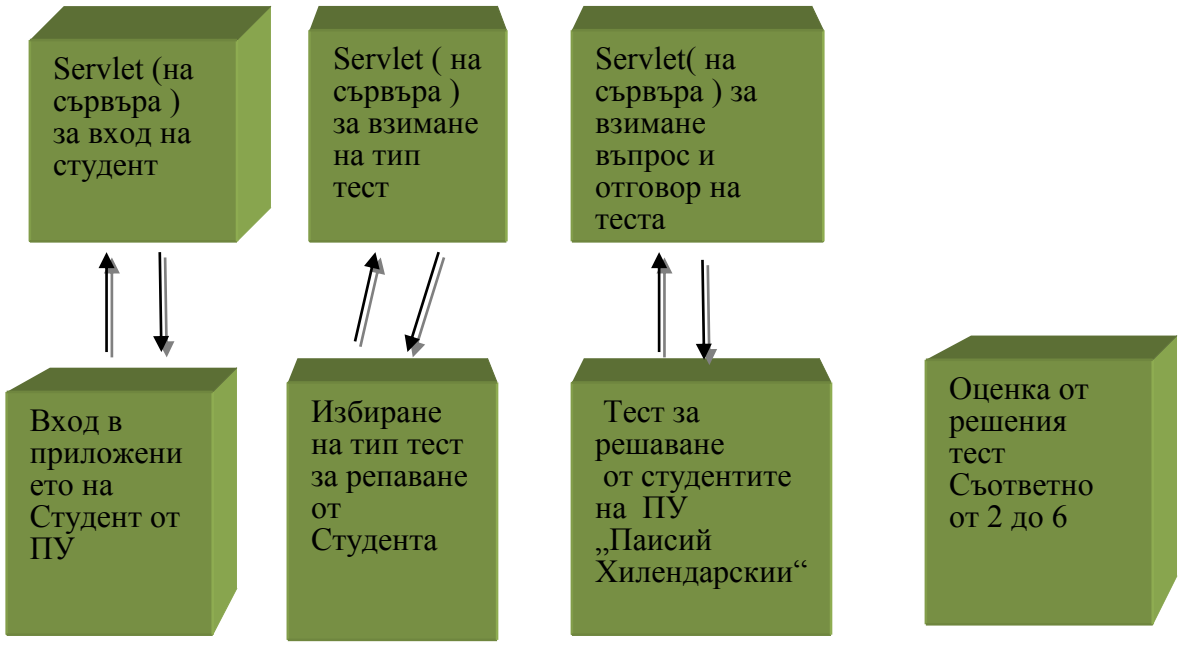
2.3 Перспективи на продукта

Мобилна тестова система е нов продукт. Той е разработен за да се използва специално от студенти от Пловдивски Университет „Паисий Хилендарскии“.

Това приложение е тип образователно приложение с което може да се проверят знанията на студентите, по избран от тях конкретен тест, и съответно на базата на знанията им се поставя оценка от 2 до 6.

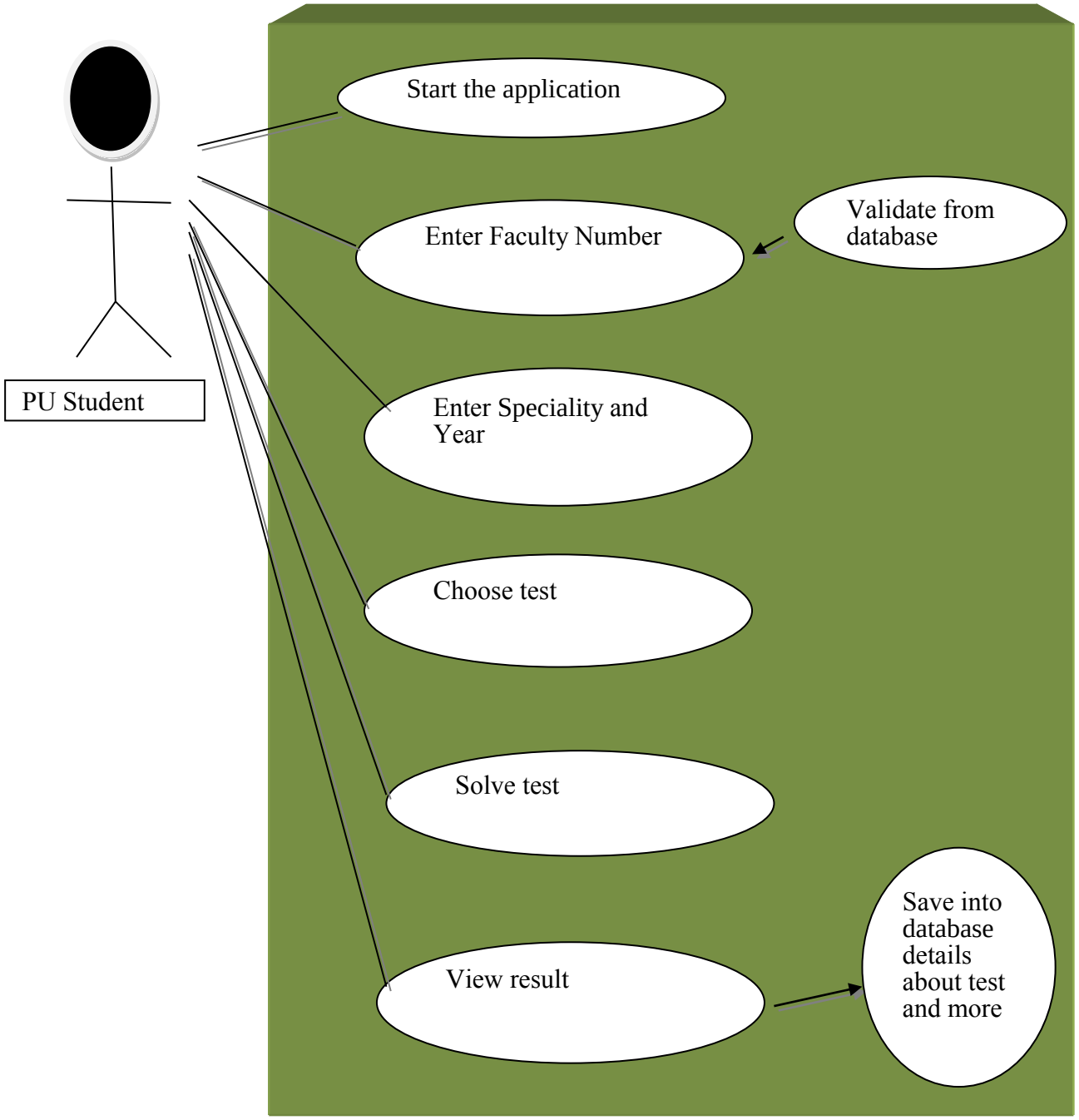
Основните компоненти на системата са показани на Фигури 1 ,2 и 3

Подобен софтуер е вече разработената Delc система за обучение на студент



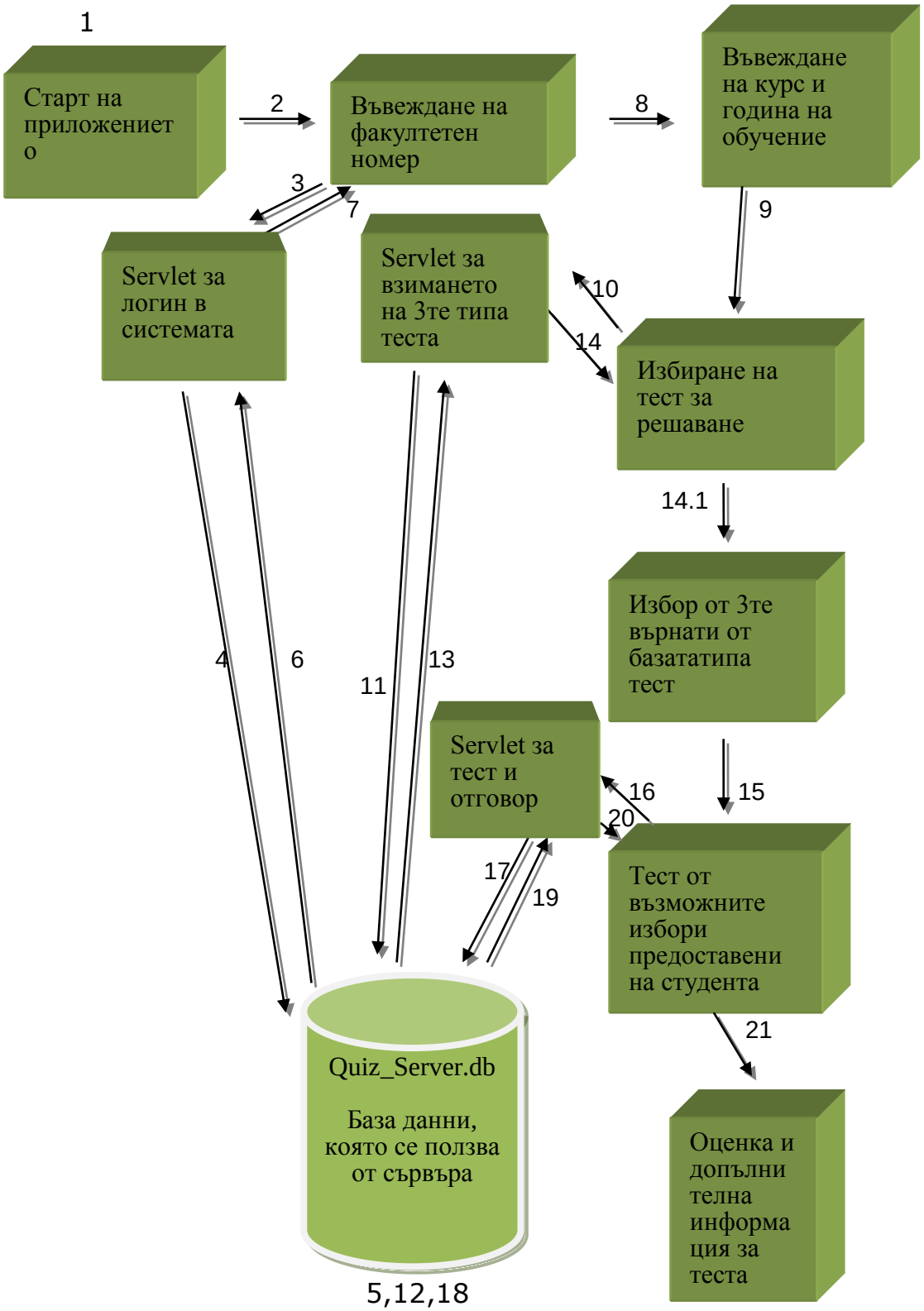
Фигура 1.

Тази фигура е кратко описание както на клиентската част, която е реализирана чрез андроидско приложение, така и на сървърната част, реализирана чрез java servlet-и. Комуникацията между андроидското приложение и servleti-те става чрез request-и(стрелката надолу) и respons-и (стрелката нагоре)



Фигура 2

Тази фигура описва какво трябва да направи един студент,чрез Use Case диаграма, за да реши един тест и да получи крайна оценка.



Фигура 3

На тази фигура е описано как клиент-сървър приложението работи по-подробно, къде и как по-точно се използват сървлетите от сървърната част на приложението. С помощта на номерата от 1до 21 са проследени стъпките на действие на ориложението

3 Използвани технологии

3.1 Андроид

Мобилният свят цъфти. Всъщност, с нарастването на търсенето на смартфони, светът също се е свил и по-малък и сега се вписва в ръката ви. Сега на пазара има разнообразие от смартфони, които се захранват от различни операционни системи. Войната в пазара на смартфони не е само между производителите на хардуер, но и между производителите на операционни системи за мобилни телефони. Google Android, Microsoft, Symbian са водещите производители на OS за смартфона.

Въпреки че има твърде много конкуренти за пазара, Android на Google е преодолял всяка друга компания и сега се е превърнал в водещ производител на операционни системи за смартфони. Това е наистина забележително за една компания да постигне това в много кратко време. Ще го намерите наистина интересно, ако знаете какво стои зад популярността на Android.

3.2 Какво е Андроид?

Андроид е операционна система, не само за смартфони, но и за таблети, телевизори и часовници наречени още wearables. Базирана е на Linux и е разработена от Open Handset Alliance консорциум от над 80 leading хардуерни компании като Sony, Samsung and и софтуерни компании, част от Google.

3.3 Версии

Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Ice Cream Sandwich (ICS) ,Marshmellow, Nouget– Това са версиите на Андроид, за различните смартфони.Nouget е последната излезнала версия и се потдържа от Samsung Galaxy S7/S7 edge.

3.4 Защо е толкова популярен?

3.4.1 Безплатен е

От момента в който излезе, Андроид ОС е безплтен и ще остане безплатен и за в бъдеще. Това решение на Google привлече вниманието на много производители на хардуер из цял свят. Тази популярност на Андроид ОС се

дължи предимно на факта, че е безплатна. Това позволява да Google да си партнира с много известни производители на хардуер.

3.4.2 С отворен код е

Андроид ОС е също с отворен код. За разлика от други Операционни системи, които са защитени от различни процеси за защита, плагиатство и други подобни, Google рещава Андроид да е отворена ОС. С това Google позволява на активните програмисти от цял свят да подобрят системата. Все повече нови и нови идеи се имплементират от умни програмисти.

3.4.3 Може да се създават собствени апликации

Ако собствениците на смартфон или таблет с Андроид ОС са заинтересовани от създаването на свои собствени апликации, това не е никакъв проблем, дори и бе никакви познания по програмиране. Google предоставя набор от инструменти, които са безплатни и позволяват на потеребителите да създават свои апликации с помощта на **Software Development Kit** за Android App Development.

3.4.4 Google Маркет

Google разпространява всички приложения за телефони с Android устройства през своя Google Play (преди Android Market). Тя позволява на потребителите да тестват, използват и споделят различни приложения и ги препоръчват на другите потребители. В магазина има над хиляди приложения. Някои от тези приложения са достъпни безплатно, а някои се заплащат. Потребителите на смартфон с Андроид ОС можете да посетят Google Play.

3.5 Предимство пред другите Операционни системи

Причините, които казах по-горе, ви разясняват защо Android е различен от другите. Всъщност не само това е достатъчно, за да бъде най-добрият играч на пазара, който побеждава тежка конкуренция. Android има големи предимства пред другите производители на операционни системи.

3.6 По – достъпен за всякакъв вид хора

Android OS е насочена към обикновения човек, който иска да получи максимални възможности на ниска цена. Дори сега много хора мислят, че iPhone е за богати. Тези обикновени мъже, които искат да имат свои собствени смартфони, могат да притежават смартфон, работещ под Android, който да е под техния бюджет.

Освен това iPhone без jailbreaking не приема много SIM, докато Android ще поддържа всяка SIM карта. Това е най-големият плюс за Android.

3.7 Restful

3.7.1 Какво означава REST?

REST означава **RE**presentational **S**tate **T**ransfer. REST е уеб базирана архитектура която използва HTTP протокол за комуникация на данни. Тя се върти около ресурси, където всеки компонент е ресурс и достъпът до ресурс се осъществява чрез общ интерфейс, използващ стандартни HTTP методи.

В архитектурата REST, един REST сървър просто осигурява достъп до ресурси, а REST клиентът достъпва и ги представ. Тук всеки ресурс се представя като URI-и/ Глобални ID-та. REST използва различни репрезентации за да представи ресурсите като Text, JSON и XML. JSON е най-популярния начин за репрезентация на данни за Уеб Сървиси.

3.7.2 HTTP Методи

Тези HTTP методи са най – често използвани в REST архитектурата.

- **GET** − Прочитане на ресурс.
- **PUT** – Използва Прехвърля ресурс към сървъра.
- **DELETE** – Унижтожава ресурс.
- **POST** − Добавя към съществуващ ресурс на сървъра.

3.8 RESTFul Web Services

Уеб услугата е колекция от отворени протоколи и стандарти, използвани за обмен на данни между приложения или системи. Софтуерните приложения, написани на различни програмни езици и работещи на различни платформи, могат да използват уеб услуги за обмен на данни чрез компютърни мрежи като интернет по начин, подобен на комуникацията между процесите на един компютър.

Тази оперативна съвместимост (например между Java и Python или Windows и Linux приложения) се дължи на използването на отворени стандарти. Уеб услугите, базирани на REST Architecture, са известни като RESTful Web Services. Тези уеб услуги използват HTTP методи за реализиране на концепцията за REST архитектура. Универсалната уеб услуга обикновено определя URI (Uniform Resource Identifier), която е услуга, която осигурява представяне на ресурси като JSON и набор от HTTP методи.

4 Разработка

В тази точка ще се разяснят клиентската и сървърната части от клиент – сървър приложението.

Всички файлове са разположени в 4 пакета(packages).

Пакетът може да бъде дефиниран като група от свързани типове, осигуряващи защита на достъпа и управление на namespace-овете. Използват се в Java, за да се предотвратят конфликтите в именуване, да се контролира достъпът, да се направи по-лесно търсене / локализиране и използване на класове, интерфейси, изброявания(enumerations) и пояснения(annotations).

4.1 Сървър

Той се състои от 4 servlet-а разположени в пакет с име servlet.

- 1-вият servlet е малко по-специален и е сложен в свой собствен пакет с име server. Той се използва за стартирането на сървъра и е с код :

```
@WebServlet("/Server")
public class Server extends HttpServlet {
    private static final long serialVersionUID = 1L;

   public Server() {

   }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

            response.setContentType("text/html; charset=utf-8");
      ConnectionDB connection = new ConnectionDB();
      if(connection.connection()){
            response.sendRedirect("Connection.jsp");
      }else{
            response.sendRedirect("Error.jsp");
      }
    }
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {

            doGet(request, response);
    }
}
```

Фигура 4

Фигурата на предишната страница представлява първи servlet за сървърната част на клиент-сървър приложението, като този 1-ви servlet се използва за пускането на сървъра,защото без да се стартира този servlet няма да работи това клиент-сървър приложение.

В servlet-а в doGet метода има инстанция connection, от тип ConnectionDB за връзка с базата данни, необходима за извличане на данните от базата данни, който после ще се достъпят от клиента чрез request. Кода за този клас е следният:

```
public class ConnectionDB {
        Connection conn = null;
        public static void main(String[] args) {

        }
    public boolean connection(){
                try{
                        Class.forName("org.sqlite.JDBC");
        conn=DriverManager.getConnection("jdbc:sqlite:/C:/Users/Sasaho/Deskto
        p/Quiz_Server.db";
                    return true;
                  }
                   catch(Exception e)
                   {
                          e.printStackTrace();
                          return false;
                   }finally{
                                try {
                                                conn.close();
                                        } catch (SQLException e) {

                                                e.printStackTrace();
                                    }
              }
            }
  }
```

Фигура 5

В този фигура се вижда един клас, в който има метод connection(),както по горе е споменато, за осъществяване на връзка с sqlite база данни с помощта на JDBC, и ако връзката е осъществена, метода в такъв случай връща true, защото той е **boolean.**

Ако обаче не се осъществи връзка с базата тогава метода връща false и съобщение при runtime за грешка. Съобщението идва от факта, че целият този процес е

заобиколен от един try{}catch() {}finaly() блок. А finaly() е използван за затваряне на връзката с базата данни (conn).

В 1-вият servlet също се среща :

```
if(connection.connection()){
                response.sendRedirect("Connection.jsp");
        }else{
                response.sendRedirect("Error.jsp");
        }
```

Фигура 6

В тази фигура се прави проверка и на базата на тази проверка се пренасочва в различни страници.

За по-подробно разяснение се прави if() {} else проверка и се ползват 2 jsp страници.

```
        <%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
                pageEncoding="ISO-8859-1"%>
                <%@ page import="java.sql.*" %>
                <%@ page import="java.util.*" %>
          <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
          Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
          <html>
        <head>
          <meta http-equiv="Content-Type" content="text/html;
          charset=ISO-8859-1">
          <title>Success</title>
          </head>
          <body>
          <%
            out.println("Database connection established");
        %>
          </body>
          </html>
```

Фигура 7

На тази фигура се вижда jsp сраницата за успешно свързване с базата данни за това приложение.

error.jsp с код:

```
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Error</title>
</head>
<body>
<%
 out.println("Database connection was not established");
%>
</body>
</html>
```

Фигура 8

На тази фигура се вижда jsp сраницата за неуспешно свързване с базата данни за това приложение.

`<%` - Типичен знак при servlet-и. Той се ползва когато програмиста иска да добави java код в jsp страница.

- 2ри servlet в пакет servlets.

```
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
     private static GsonBuilder gson_builder = new
GsonBuilder().serializeNulls().setDateFormat("MM/dd/yyy
y");
        private UserService userService;

   public LoginServlet() {
      super();
      userService = new UserService();
   }
```

Фигура 9

Тази фигура показва част от LoginServlet за студентите, като останалите 2 части от servlet-а са представени в следващите 2 фигури.

По-долу ще бъде представен и doGet метода.

```
    protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
                    String password =
request.getParameter("facNum");
            User user = userService.find(password);
            if (user != null) {

response.setContentType("application/json;charset=UTF-8");
                Gson gson = gson_builder.create();
                response.getWriter().write(gson.toJson(user));
            }
            else {
                request.setAttribute("error", "Unknown user,
please try again");
            }
    }
```

Фигура 10

На тази фигура е представен doGet метода, в който цялата логика е изнесена, и за който се споменава в предишната фигура.

А doPost не се ползва.

```
    protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException,
IOException {
    }
}
```

Фигура 11

На тази фигура е показан doPost метода като той е абсолютно празен поради простата причина, че не се ползва за никаде.

```
public User getUser(String facNum) {
User user = new User();
    return user;
  }
```

Фигура 12

На тази фигура се вижда User класа, който е в пакет objects и е обект, който просто държи информацията за логнатият в програмата студент (по- точно в

клиентското приложение). Той е в класа ConnectionDB за връзка с базата данни за сървърната част на приложението.

UserService е клас в пакет services и има за цел да използва базата данни:

```
public class UserService {

        public User find(String facNum){
                ConnectionDB connection = new
        ConnectionDB();
                    if(connection.connection()){
                            User user =
        connection.getUser(facNum);
            }
```

Фигура 13

На тази фигура се вижда как ако няма студент специално с този факултетен номер, тогава се прави ново търсене в базата данни и се изкарва цялата необходима информация за студента.

```
        User user = new User();
          Connection conn = null;
           Statement stmt = null;
           try{
                Class.forName("org.sqlite.JDBC");
          conn=DriverManager.getConnection("jdbc:sqlite:/C:/Users/Sas
aho/Desktop/Quiz_Server.db");
              conn.setAutoCommit(false);
              stmt = conn.createStatement();
              ResultSet rs = stmt.executeQuery("SELECT * FROM
Student_table where FacultyNumber LIKE " + facNum );
              while (rs.next()){
                 user.name = rs.getString("FirstName");
                 user.family = rs.getString("LastName");
                 user.facNum = rs.getString("FacultyNumber");
              }
           }
           catch(Exception e){
                e.printStackTrace();
           }
                return user;
        }
```

Фигура 14

Фигура 14 показва, ако User не е намерена с помощта на горния метод се порверява в базата данни и се попълва информацията за него, споменато по-гор.

Малко по-подробно за UserService. Той е помощен клас, който се използва за операции, свързани с базата данни(като ако пожелаем може да не се ползва този клас, а директно да се ползва базата данни, но тук се ползва, за separation of concerns и също така, ако се обърка нещо в кода, после да се ориентираме по-лесно къде е проблемът)

Като за целта, за да бъде използван се прави един gson_bulder,

```
private static GsonBuilder gson_builder = new
GsonBuilder().serializeNulls().setDateFormat("MM/dd/yyyy");
```

Фигура 15

На тази фигура се показва 3rd party library от google, както беше споменато по-горе, наречена gson, за създаването на json обект (Веднага щом се получи необходимата информация от базата данни те се подават на gson за създаване на json обект) . Тук нарочно е избрано да се работи с content type json вместо с xml, поради простата причина, че е по-лесно четим от xml.

Нататък ще се погрижи за преобразуването на получения json като request и отново с gson, но в клиента, ще се обърне в използваем бект.

```
Gson gson = gson_builder.create();
response.getWriter().write(gson.toJson(user));
```

Фигура 16

Фигурата показва как gson_builder се ползва за създаването от обект (в този сучай е user) в json, като java ползва json за удобство, но естествено java може да ползва и xml. (това вече бе споменато)

- 3-ти servlet в пакет servlets.

Той е предназначен специално за избор на тест.

Като възможността за избора на един от 3-те възможни варианта (Android_test, Java_test, CSharp_test) е предоставена отново на клиентското приложение, а вече взимането на тези три възможни теста се случва с помощта на този 3-ти servlet.

```
@WebServlet("/TestTypeServlet")
public class TestTypeServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static GsonBuilder gson_builder = new
GsonBuilder().serializeNulls().setDateFormat("MM/dd/yyyy");
        private TestTypeService testTypeService;
public TestTypeServlet() {
        super();
        testTypeService = new TestTypeService();
    }
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
            TestTypes typeOfTest = testTypeService.getTestType();
        response.setContentType("application/json;charset=UTF-8");
        Gson gson = gson_builder.create();
        response.getWriter().write(gson.toJson(typeOfTest));
    }
   protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
                        doGet(request, response);
    }
 }
```

Фигура 17

На тази фигура е представен кода за 3-ти servlet за приложението като по-горе е обяснено също за какво се използва той. Допълнително бих добавил, че servlet-ите (от 2 до 4) играят ролята на response.

За целта на този servlet се използва TestTypes клас с код :

```
public class TestTypes {
    public static final String TEST_TYPE_JAVA = "JavaQuestion_table";
    public static final String TEST_TYPE_C_SHARP
="CSharpQuestion_table";
    public static final String TEST_TYPE_ANDROID = "Question_table";

    public String[] typeTests = new String[]{
                TEST_TYPE_ANDROID,
            TEST_TYPE_C_SHARP,
            TEST_TYPE_JAVA
        };
}
```

Фигура 18

На горната фигура 18 се вижда как в прост масив се записват 3-те типа тест, които по късно ще се вземат от клиентското приложение с request.

TestTypeService поставен в пакет services с код:

```
public class TestTypeService extends TestTypes {

    public  TestTypes getTestType()
     {
            return new TestTypes();
        }
}
```

Фигура 19

На тазу фигура е показан Service, който има само един метод с име getTestType(). Неговата цел е да връща трите типа теста.

- 4-ти servlet отговаря за взимането на отговорите на генерираните на свободен принцип въпроси за тестове, които вече са споменати по-горе (тук пак ги припомням споменем те са Java_test, Android_test, CSharp_test).

По-конкретно всеки от тези тестове има 3 вида въпроси.

```
@WebServlet("/QuestionAndAnswerServlet")
public class QuestionAndAnswerServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private static GsonBuilder gson_builder = new
GsonBuilder().serializeNulls().setDateFormat("MM/dd/yyyy");
     private QuestionService questionService;
   public QuestionAndAnswerServlet() {
      super();
questionService = new QuestionService();
   }
    protected void doGet(HttpServletRequest request,
 HttpServletResponse response) throws ServletException, IOException
 {
       String testType = request.getParameter("testType");

            Question questionAndAnswer =
 questionService.getQuestionAndAnswerFromDB(testType);
             if(questionAndAnswer!=null){
             response.setContentType("application/json;charset=UTF-
 8");
       Gson gson = gson_builder.create();
    response.getWriter().write(gson.toJson(questionAndAnswer));
             }
```

Фигута 20

На фигура 20 на предишната страница се представя servlet-а, който е отговорен за взимането на генериран на свободен принцип въпрос, както и последвания отговор за въпроса като цялата логика отново е изкарана в doGet метода.

```
        protected void doPost(HttpServletRequest request,
    HttpServletResponse response) throws ServletException, IOException
    {
    }
}
```

Фигура 21

Тук на тази фигура е показана другата част от servlet-а, именно doPost метода, който отново не се използва.

Освен servlet-а се ползва QuestionService,Question,Answer и 3rd party library от google.

Код за Question класа:

```
public class Question {
public int id;
public String questionText;
public int type;
public List<Answer> answers;
}
```

Фигура 22

Тази фигура показва клас за обект (Question), който просто държи цялата информация за въпроса, който ще бъде зададен на студента, казано по друг начин дава информация за полета свързани с въпроса(като Id на въпроса, текста на въпроса, type на въпроса и един List<Answer> answers свързани с върпоса или още познато като модел на Question_table,CSharp_Question_table,Java_Question_table като и 3те са с еднаква структура).

Също съществува метод от тип Question в класа ConnectionDB за връзка с базата данни с код:

```
public Question getQuestionAndAnswerFromDB(String nextQuestion){
        Question question = new Question();
        return question;
    }
```

Фигура 23

На фигура 23 се покава метод getQuestionAndAnswer поставен отново в ConnectionDB за връзка с базата данни на приложението.

```
public class Answer {
        public int id;
        public int questionId;
        public String answerText;
        public boolean isCorrect;
}
```

Фигура 24

Тази фигура показва Answer класа. Той пък е репрезентация на един цялостен отговор(или още познато като model на Answer_table,CSharpAnswer_table, JavaAnswer_table, като и 3те отново са с еднаква структура) и QuestionService в пакет services

Споменатият по-горе метод с респективният за него код:

```
private Question getRandomQuestion(String testTableName){
                goToDatabaseLocation();
                try{
                        Question question = new Question();
                        ResultSet rsQuestion =
    stmt.executeQuery("SELECT * FROM "+ testTableName + " order by
    random() limit 1");
                    while (rsQuestion.next()){
                    question.id = rsQuestion.getInt("ID");
                    question.questionText =
    rsQuestion.getString("Question");
                    question.type =  rsQuestion.getInt("Type");
                  }
                    return question;
                }catch(Exception e){
                    e.printStackTrace();
                }
                    return null;
            }
```

Фигура 25

На фигура 25 се В този метод се вижда как на базата на подаден тип тест в URL-а се взима генериран въпрос на произволен принцип (като за него се взима цялата информация от базата данни)

```
private List<Answer> getAnswersByQuestionId(String testType, int
questionId) {
        List<Answer> answers = new ArrayList<Answer>(3);
        String answerTableName= null;
        switch (testType) {
        case "Question_table":
                answerTableName = "Answer_table";
                break;
         case "CSharpQuestion_table":
                answerTableName = "CSharpAnswer_table";
                break;
         case "JavaQuestion_table":
                answerTableName = "JavaAnswer_table";
                break;
              default:
                break;
              }
```

Фигура 26

Тук се представя началото на един метод. На базата на типа на тест се определят и нужните отговори, които ще бъдат базирани на въпросите зададени на студент.

```
        try{
                ResultSet rsAnswer = stmt.executeQuery("SELECT * FROM
" + answerTableName +" WHERE QuestionID = " + questionId );
            while (rsAnswer.next()){
              Answer answer = new Answer();
              answer.id = rsAnswer.getInt("AnswerID");
                  answer.answerText = rsAnswer.getString("Answer");
                  answer.questionId = questionId;
                  answer.isCorrect = rsAnswer.getBoolean("IsCorrect");
                  answers.add(answer);
            }
           }catch(Exception e){
                e.printStackTrace();
           }
        return answers;
}
```

Фигура 27

Съответно тук на тази фигура се показва как се взимат отговорите от базата данни, и се разпределят в switch() case {} по–горе в началото на метода.

4.2 Клиент

Преди да се опише клиентското приложение на бързо ще се спомене какво е използвано за клиентското приложение.

То се състои от :

- Edit text - ове (widget) за полета, които се попълват от студентите
- Text View – та (widget) за въпросите
- Кръгъл (custom made) Progress Bar, за да следи за оставащото време
- Използвани са също custom TOAST съобщения за информация
- Scroll View за по лесно разглеждане на цялото съдържание на страницата
- Custom Background за Edit text – овете
- Vertical Layout – 3 на брой
- Радио група
- CheckBoxes – за multipleAnswer отговори
- Радио бутони – за singleAnswer отговори
- Rich Text Area – като тя всъщност е един editText
- Добавен е и custom ripple_effect за бутоните
- Custom направена сянка – тя е направена отново чрез xml файл.
- Custom изображения – те са обработвани с програмата Photoshop за да служат за дизайна на приложението

След като е споменато какво ще се използва за дизайна на клиентското приложение.

Те са разделени в отделни папки – model, async, android, ui. Папката ui се състои от StartTestActivity клас за стартирането на тест. Още в началото на StartTestActivity глобално се използват няколко променливи

```
ImageButton imgStartTest;
DatabaseHelper MyDb;
EditText etFacNumber;
ImageButton imgLogin;
TextView tvWelcome, checkFacultyNumber, tvSpecialityAndYear;
Spinner optionSpinner;
```

Фигура 28

На тази фигура sa показани някои променливи, като те са дефинирани още в самото начало, за да се виждат и използват от цялото Activity

```
ArrayAdapter<CharSequence> adapter;
  String testType, fullName;
  EditText specialityAndYear;
  private final Gson gson = new GsonBuilder().create();
```

Фигуеа 29

На тази фигура е показано дефинирането на някои допълнителни променливи заедно с горните променливи от фигура 28 като те са дефинирани още в самото начало, за да се виждат и използват от цялото Activity

С помощта на onCreate метода:

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_start_test);

   Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
   toolbar.setTitle("Start of the test");
   toolbar.setTitleTextColor(Color.WHITE);
   setSupportActionBar(toolbar);

   checkFacultyNumber = (TextView) findViewById(R.id.tvFacultyNumber);
   tvSpecialityAndYear = (TextView) findViewById(R.id.tvSpecialityAndYear);
   specialityAndYear = (EditText)
findViewById(R.id.specialityAndYearEditText);
   imgStartTest = (ImageButton) findViewById(R.id.StartTest);
   imgStartTest.setVisibility(View.INVISIBLE);
   buttonEffect(imgStartTest);
   imgLogin = (ImageButton) findViewById(R.id.ibStart);
   etFacNumber = (EditText) findViewById(R.id.studentEditText);
   tvWelcome = (TextView) findViewById(R.id.tvWelcome);
   optionSpinner = (Spinner) findViewById(R.id.optionSpinner);

   imgLogin.setOnClickListener(onLoginClickListener);
   imgStartTest.setOnClickListener(start);
   optionSpinner.setVisibility(View.INVISIBLE);
}
```

Фигура 30

На фигура 30 е показн onCreate метод, чиято логика на работа се преповтаря и в други 2 Activity-та за клиентското приложение. Съответната логика, както е казано по-долу е да референцира променливите за по нататъчна употреба. Като освен това се „закача” onClickListener натискането на конкретните бутони (`imgLogin,imgStartTest`).

```
Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);

   toolbar.setTitle("Start of the test");

   toolbar.setTitleTextColor(Color.WHITE);

   setSupportActionBar(toolbar);
```

Фигура 31

На фигурата е описано как по-точно става задаването на ToolBar за клиентското приложение.

```
View.OnClickListener onLoginClickListener = new View.OnClickListener() {
   @Override
   public void onClick(View v) {

      if (TextUtils.isEmpty(etFacNumber.getText().toString())) {
         etFacNumber.setError("Полето не е попълнено");
      } else {
         login();
      }
   }
```

Фигура 32

Тази фигура описва по-подробно единият от по-горе споменатите liatener-и за това клиентско приложение. Тук се проверява дали е празно полето и ако да, тогава се преминава към метод login()

```
public View.OnClickListener start = new View.OnClickListener() {
   public void onClick(View v) {
      if (TextUtils.isEmpty(testType))
      {
      }
 else
 {
    startActivity(QuestionActivity.newIntent(StartTestActivity.this, testType));
  }
 }
```

Фигура 33

На фигура 33 е представен другият listener за старт на избран от студента тест(евентуално ако е избран някакъв) и преминаване в чисто ново Activity за теста

```
private void login() {
    new LoginAsyncTask(new ApiCallback() {
        @Override
        public void onResponse(String response) {
            UserResponse userResponse = gson.fromJson(response, UserResponse.class);
            if (userResponse==null) {
                showError("Server error");
            } else if(TextUtils.isEmpty(userResponse.name) ||
userResponse.name.equalsIgnoreCase("null")) {
                showError("Wrong credentials");
            } else {
                tvWelcome.setText(userResponse.name + " " + userResponse.family);
                fullName = tvWelcome.toString();
                getTestTypes();
            }
        }
    }).execute(etFacNumber.getText().toString(),fullName);
}
```

Фигура 34

Тази фигура описва кода свързан с извикания login() метод показан в първият listener на фигура 32.

Тук съотвтно се ползва LoginAsyncTask от папката async описан по-долу във фигура 37 и фигура 38.

Допълнително се ползва gson библеотеката която има за цел да преобразува полученият response до обикновен обект за по-нататъчно ползване.

```
public class UserResponse {
    public String name;
    public String family;
    public String facNum;
}
```

Фигура 35

На тази фигура е показан класа UserResponse. Това е класа които се подава на gson заедно с response-а, който се пребразува в конкретният UserResponse клас.

```
public LoginAsyncTask (ApiCallback apiCallback){
    this.apiCallback = apiCallback;
}
```

Фигура 36

На фигура 36 се описва ApiCallback, който е част от LoginAsyncTask. Той от своя страна се извиква за ползване в метод login() .

```
protected String doInBackground(String... params) {
    URL url;
    HttpURLConnection urlConnection;
    BufferedReader br;
    try{
        url = new URL(LOGIN_URL + params[0] );
        urlConnection = (HttpURLConnection) url.openConnection();
        br = new BufferedReader(new InputStreamReader(
                urlConnection.getInputStream()));
        StringBuilder sb = new StringBuilder();
        String line = br.readLine();
        while (line != null){
            sb.append(line);
            line = br.readLine();
        }
        return sb.toString();
    }catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
```

Фигура 37

На тази фигура се описва метод doInBackground() в LoginAsyncTask на това клиентско приложение, също така освен този метод има и още един метод, като за него ще се спомене малко по-нататак.

По-подробно разяснение за ApiCallback.

```
public interface ApiCallback {
    void onResponse(String response);
}
```

Фигура 38

Тук на тази фигура се показва отново ApiCallback, като е обяснено по-подробно какво представлява той.

Той е интерфейс изпращащ request до сървъра, като сървъра връща response. Този response после се използва от gson библеотеката,която вече за клиента преобразува response-а в обект за ползване.

Тази цялата логика е обясена отново във фигура 40 и фигура 41.

```
public class ApiConstants {
   public static final String URL = "http://192.168.0.104:8080/Server/";
}
```

Фигура 39

Тази фигура показва ApiConstants клас. Като между скобите се задава съответната част от URL-а, обща за приложението, а по-късно в AsyncTask-овете ще се ползва за дописванетон на URL- а, необходим за взимането на данните от сървъра.

```
private void getTestTypes() {
   new TypeTestsAsyncTask(new ApiCallback() {
      @Override
      public void onResponse(String response) {
         TestTypes testTypes = gson.fromJson(response, TestTypes.class);
         adapter = new ArrayAdapter(StartTestActivity.this,
android.R.layout.simple_spinner_item);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
         adapter.addAll(testTypes.typeTests);
         optionSpinner.setAdapter(adapter);
         optionSpinner.setVisibility(View.VISIBLE);
         imgStartTest.setVisibility(View.VISIBLE);
         setOnTestTypeSelectedListener();
      }
   }).execute();
}
```

Фигура 40

Във фигура 40 е представен метода getTestTypes().

Той сам по себе си използва вторият asyncTask в папката async с име TypeTestsAsyncTask, който в метода си doInBackground прави request до сървъра (като се има в предвид, че в сървъра се взимат всички 3 типа тест и след това те се връщат като response на request-а от приложението. След това и 3те теста се пълнят в adapter, а той се подава на Spinner с име optionSpinner)

Като отново се използв gson със същата идея (да преобразува respons-а до код, който после да може да се използва от приложениет.

```
TestTypes testTypes = gson.fromJson(response, TestTypes.class);
```

Фигура 41

На фигура 41 е показано как response-а получен от сървърната част се преобразува в обект с име TestTypes сепециално с използването на gson библеотеката на google.

Интересното тук е между двата doInBackground() метода в двата asyncTask-а. Единият подава String params за request-а (LoginAsyncTask), а другият не подава никакви параметри, освен че ползва нужният му URL (TypeTestsAsyncTask).

```
private void setOnTestTypeSelectedListener() {
   optionSpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener()
{
      @Override
      public void onItemSelected(AdapterView<?> parent, View view, int position, long
id) {

         ((TextView) parent.getChildAt(0)).setTextColor(Color.WHITE);
         testType = (String) optionSpinner.getSelectedItem();
      }
```

Фигура 42

На фигура 42 е покзанао как студент избира един от 3те теста за решавне като този listener се извиква в asyncTask-а в login метода преди да се execut-не той.

Следващото Activity в папка ui е QuestionActivity, то е доста подобно на StartTestActivity.

Метода onCreate() отново се използва за референцирането на променливи, които ще се ползват.

За допълнение се викат и методи като : `setTitle();`

```
private void setTitle() {
   switch (getExtraTestType()) {
      case ANDROID_TEST_TYPE:
         getSupportActionBar().setTitle("ANDROID TEST");
         break;
      case CSHARP_TEST_TYPE:
         getSupportActionBar().setTitle("CSHARP TEST");
         break;
      case JAVA_TEST_TYPE:
         getSupportActionBar().setTitle("JAVA TEST");
        break;
   }
}
```

Фигура 43

На фигура 43 е показано как на базата на избрания тест идващ от екстрата с име getExtraTestType() се задава заглавие за TitleBar-а.

Метод initTimer();

```
private void initTimer() {
    app.setCountdownListener(this);
    pbTimer.setMax((int) MainApplication.TEST_DURATION_IN_MILLIS);
 }
```

Фигура 44

На тази фигура е показано стартирането на Timer за теста, който се решава от студент в клиентското приложение.

Като логиката за този Timer е изнесена в MainApplication клас, а CountdownListener е интерфейс с метод **void** timerTick(**long** millisUntilFinished);

който трябва задължително да бъде имплементиран, за да се ползва listener-а.

```
@Override
public void timerTick(long millisUntilFinished) {
   tvTimer.setText(String.format("%d : %d",
         TimeUnit.MILLISECONDS.toMinutes(millisUntilFinished),
         TimeUnit.MILLISECONDS.toSeconds(millisUntilFinished) -
TimeUnit.MINUTES.toSeconds(TimeUnit.MILLISECONDS.toMinutes(millisUntilFini
shed))));

   pbTimer.setProgress((int) millisUntilFinished);
}
```

Фигура 45

Тази фигура показва имплементираният метод от интерфейса CountdownListener, тъй като, за да се използва интерфейс трябва задължително да се имплементират неговите методи, както по-горе е споменато

Метод readQuestionAndAnswerFromServer(); за взимането на въпроси и отговори от сървъра, като тук се ползва последният AsyncTask, за да се вземат те от сървъра.

Метод animateProgressBar();

```
public void animateProgressBar() {
    ProgressBar mProgressBar = (ProgressBar)
findViewById(R.id.progressBarFirstQuestion);
    anim = ObjectAnimator.ofInt(mProgressBar, "progress", 100, 0);
    anim.setDuration(MainApplication.TEST_DURATION_IN_MILLIS);
    anim.setInterpolator(new DecelerateInterpolator());
    anim.start();
}
```

Фигура 46

Фигура 46 описва по-подробно анимиране на progressBar-а за Timer-а на клиентското приложение за QuestionActivity.

```
private void readQuestionAndAnswerFromServer() {
    new QAAsyncTask(new ApiCallback() {
        @Override
        public void onResponse(String response) {
            currentQuestion = gson.fromJson(response, Question.class);
            showQuestionLayout(currentQuestion);
        }
    }).execute(getExtraTestType());
}
```

Фигура 47

На тази фигура е описан методът readQuestionAndAnswerFromServer(), който по-горе беше споменат, но не и описан. Този метод използва и последният AsyncTask в папка async с име QAAsyncTask, като този AsyncTask е идентичен с LoginAsyncTask, само че в doInBackground се извършва request за въпросите и отговорите на теста. Respons-а тук отново идва под формата на json и за това се ползва библеотеката на google, gson, която има за задача да преобразува този response в готов обект за използване в папка model:

```
public class Question {
    public int id;
    public String questionText;
    public int type;
    public List<Answer> answers;
}
```

Фигура 48

На фигура 48 се описва един клас, който преминава през gson, за да стане от json response в обект, който да може да се изпползва.

С метод toString() :

```
@Override
    public String toString() {
     return "Question{" +
          "id=" + id +
          ", questionText='" + questionText + '\'' +
          ", type='" + type + '\'' +
          ", answers=" + answers +
          '}';
    }
```

Фигура 49

На тази фигура е показан метод toString в обект Question, целта поради която се ползва метода е, за да се получи съответния въпрос, а не някакви непознати символи. Този клас се прави от програмиста ръчно с цел след това gson библеотеката да преобразува полученият string (response) обратно в обект.

Подобен клас е класа Answer, който отново е в папката model:

```
public class Answer {
  public int id;
  public int questionId;
  public String answerText;
  public boolean isCorrect;
}
```

Фигура 50

На тази фигура се описава клас, който е много подобен на Question класа, само че вместо за въпроси, както Question той се занимава с отговорите на тези въпроси.

С метод toString(), който е показан в следващата страница на документацията, за да може да се побере и да не пречи на долният колонтитул.

```
@Override
  public String toString() {
    return "Answer{" +
        "id=" + id +
        ", questionId=" + questionId +
        ", answerText='" + answerText + '\'' +
        ", isCorrect=" + isCorrect +
        '}';
  }
}
```

Фигура 51

Тук, на тази фигура се описва отново методът toString, поради абсолютно същите причини които бяха споменати в метода toString , но отнасящи се за Question

Метода с код:

```
  private void setRadioGroupSelectedListener() {
    rgAnswers.setOnCheckedChangeListener(new OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(RadioGroup group, int checkedId) {
        tvStatus.setText("Отговор бе избран!");
        tvStatus.setTextColor(Color.GREEN);
        btnNextQuestion.setEnabled(true);
        btnNextQuestion.setAlpha(0.5f);
      }
    });
  }
```

Фигура 52

На тази фигура се показва как се валидира бутона за следващ въпрос от гледна точка на радио група-та, самият горепосочен метод се вика в метод с име showQuestionLayout ()

В този метод(showQuestionLayout()) се попълват 3-те layout-а за 3-те отговора. Като 3-те layout-а репрезентират единичен правилен отговор, множество от правилни отговори и отговор със свободен текст.

На базата на типа на Question –а преобърнат в обект от от gson библеотеката се пуска switch () case {} съответно за 3-те отговора.

Подобно както има setRadioGroupSelectedListener има и setCheckBoxesChangeListeners(), показан на долната страница.

```
private void setCheckBoxesChangeListeners() {
   chOne.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)
{
         enableNext();
      }
   });
}
```

Фигура 53

Тази фигура показва как се закача setCheckBoxesChangedListener специално за checkbox, като за другите два checkbox-а е абсолютно идентично като логика.

В метода с име enableNext()

се случва реалната валидация, която позволява да се отиде на друг (генериран на случаен принцип) въпрос.

```
public void onTextChanged(CharSequence s, int start, int before, int count) {
   if (count == 0) {
      tvStatus.setText("Все още не е въведен текст!!");
      tvStatus.setTextColor(Color.RED);
      btnNextQuestion.setEnabled(false);
      btnNextQuestion.setAlpha(0.1f);
   } else {
      tvStatus.setText("Вече е въведен текс!");
      tvStatus.setTextColor(Color.GREEN);
      btnNextQuestion.setEnabled(true);
      btnNextQuestion.setAlpha(0.5f);
}
```

Фигура 54

На фигурата е показано част от addFreeTextChangeListener(); и по-скоро onTextChanged ()

Като на **etFreeAnswer**.addTextChangedListener(**new** TextWatcher()

се закача TextChangedListener и на onTextChanged отново се валидира дали е въведено нещо.

Listener с име nextQuestionOnClickListener обвързан с показването на следващ въпрос за студента. (за жалост кода не е поместен тук защото е прекалено голям)

Този listener се грижи и за раздаването на точки съответно за избран правилен отговор. Като тези точки се раздават на базата на 3-те типа възможни отговори чрез

един switch() case {} на базата на типа на въпроса и се предават като екстра за финалното Activity:

```
moveToEnd.putExtra("Grade", sumPoints);
```

Фигура 55

Тази фигура показва поставянето на екстра, която се ползва за финалното Activity и по-точно за сформиране на крайна оценка.

```
if (counter == 6) {
   Intent moveToEnd = new Intent(QuestionActivity.this, EndTestActivity.class);
   moveToEnd.putExtra("Grade", sumPoints);
   startActivity(moveToEnd);
}
```

Фигура 56

На тази фигура е показано как този (nextQuestionOnClickListener()) listener се ползва за стартирането на финалното Activity, след като се отговори на точно 6 произволно избрани въпроса

Други методи в това Activity

```
private void populateRadioButtons(Question question) {
   rbOne.setText(question.answers.get(0).answerText);
   rbTwo.setText(question.answers.get(1).answerText);
   rbThree.setText(question.answers.get(2).answerText);
}
private void populateCheckBoxes(Question question) {
   chOne.setText(question.answers.get(0).answerText);
   chTwo.setText(question.answers.get(1).answerText);
   chThree.setText(question.answers.get(2).answerText);
}
```

Фигура 57

Тази фигура показва попълването на текста (setText) на radio buttons респективно за първият метод и за check boxes за вторият, а за text area-та съответно няма такъв метод, защото както подсказва името отговорът ще бъде въведен от студента.

Съществува още един метод:

```
public void onBackPressed() {
   Toast.makeText(QuestionActivity.this, "BackPressButton Disabled!",
Toast.LENGTH_SHORT).show();
}
```

Фигура 58

На горната фигура е показан как се имплементира onBackPressed()

Той е направен нарочно по този начин, когато тест бива пуснат да не може да се върне назад и да се избере друг. Единсственият начин, по който може да се спре теста е да бъде решен до край.

Както споменах по-горе в listener-а за клика на бутона след 6-тият въпрос се преминава в ново Activity е с име EndTestActivity.

Тук ще опиша методите които са по-специфични за него, а именно getWholeGrade() moreInfoAboutTest, който е listener за единият от двата бутони и closeApplication, който отново е listener специално за затваряне на целият тест.

Отново се използва метода onCreate(), където в него се закачат няколко listener-и Съответно за image Button с име closeApp и Image Button sumAnswers:

```
sumAnswers.setOnClickListener(moreInfoAboutTest);
closeApp.setOnClickListener(closeApplication)
```

Фигура 59

На тази фигура се показва как се сакача clickListener за двата бутона, които се използват във финалното Activity.

getWholeGrade() методът има за цел да показва скалата за оценяване:

```
    private void getWholeGrade(){
        wholeGrade = getIntent().getDoubleExtra("Grade",grade);
        if(wholeGrade>25.5 && wholeGrade<=30){
            grade=6;
        }else if(wholeGrade>20.5 && wholeGrade<=25.5){
            grade = 5;
        }else if(wholeGrade>15.5 && wholeGrade<=20.5){
            grade = 4;
        }else if(wholeGrade>=10.5 && wholeGrade<=15.5){
            grade = 3;
        }else{
            grade = 2;
        }
       Toast toast = Toast.makeText(EndTestActivity.this, "Оценка от теста: " +
    String.valueOf(grade), Toast.LENGTH_SHORT);
        View toastView = toast.getView();
        toastView.setBackgroundResource(R.drawable.toast_message_style);
        toast.show();
    }
```

Фигура 60

На горната фигура 60 се показва как на базата на взетата екстра със събраните точки от QuestionActivity и отново чрез custom тост се показва вече оформена оценка по скала за оценяване.

moreInfoAboutTest е един listener, както споменах вече:

```
View.OnClickListener moreInfoAboutTest = new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        moreInformation = (TextView)findViewById(R.id.tvMoreInfo);
        moreInformation.setText("Скала за оценяване: \nпод 10.5 точки 2\n"
+
                "от 10.5 до 15.5 точки: 3\n" +
                "от 16.5 до 20.5 точки: 4\n" +
                "от 21.5 до 25.5 точки: 5\n" +
                "от 26.5 до 30 точки: 6\n");

    }
};
```

Фигура 61

На фигура 61 се показва как при натискане на бутона sumAnswers се показва на екрана едно TextView за информация на студента със скалата използвана за оценяването на теста.

```
View.OnClickListener closeApplication = new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        Toast toast = Toast.makeText(EndTestActivity.this, "Android test ще
се затвори!", Toast.LENGTH_SHORT);

          View toastView = toast.getView();
          toastView.setBackgroundResource(R.drawable.toast_message_style);

          toast.show();

          finishAffinity();
      }
  };
```

Фигура 62

На фигура 62 се показва как се използва един listener за излизането от тест при натискането на бутон за излизане(closeApplication) от клиентското приложение
Като се показса custom направен тост и се затваря приложението чрез метода finishAfficiently().

4.3 Функции на продукта

Въпросите за тестовете на приложението са генерирани на произволен принцип.

Добавен е таймер, който следи за определено време относно изпълнението на теста.Той предупреждава студентите преди да им изрече времето.

Може да са използва от всяко устройство което използва Android операционна система. Още от самото начало е лесно за използване на основни елементарни функции.Бутоните трябва да са достатъчно големи и да се видими за потребителите, освен за хора с отлично зрение, така и за хора със затруднено или проблемно зрение

4.4 Изисквания за безопасност

От гледна точка на безопасността, не са допуснати грешки или бъгове в софтуера, но ако съответно някъде се получи бъг, трябва да се уведоми за него и в следваща версия на приложението този бъг евентуално ще бъде отстранен от приложението.

4.5 Изисквания за сигурност

От гледна точка на сигурността на данните те ще могат да се виждат само и единствено от легални студенти на Пловдивския Университет. Това ще рече, че данните ще са достъпни само и единствено за студенти, чиито факултетни номера вече присъстват в официалната база данни на Пловдивския Университет.

Естествено данните няма да са видими изобщо за хора извън Университета. Това ще рече, че студенти от други университети няма да имат достъп до това приложение.

5 Документация за клиента

Приложението е достатъчно добре направено, така че потребителите му да могат да се ориентират как да работят с него без да име е необходима допълнителна документация как да си решат теста в приложението.

Но за по лесна употреба ще бъде добавен един User Guide , ще се представи по – долу в документацията и който може да се използва от клиентите при евентуален проблем, така и от разработчиците на Андроид приложения за по – голяма яснота.

Наръчник за потребителя

За този „Наръчник на потребителя" се предполага, че клиентите вече са се сдобили с apk файла на приложението.

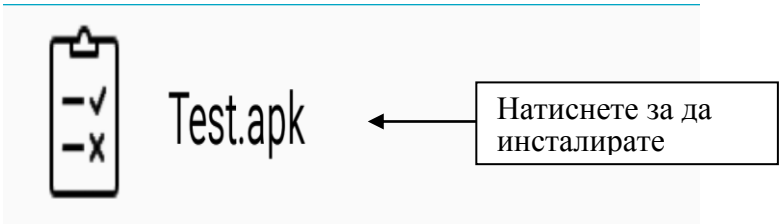
Ако ли не тогава те могат да се сдобият с файла от github с git bash, като се изплзва командата git clone.

За линк се използва - https://github.com/sashe944/apkFile

А от гледна точка на разработчиците цялото приложение ще бъде достъпно в същото repository показано по горе за да си направят clone repository и да имат достъп до целия код на своята машина.

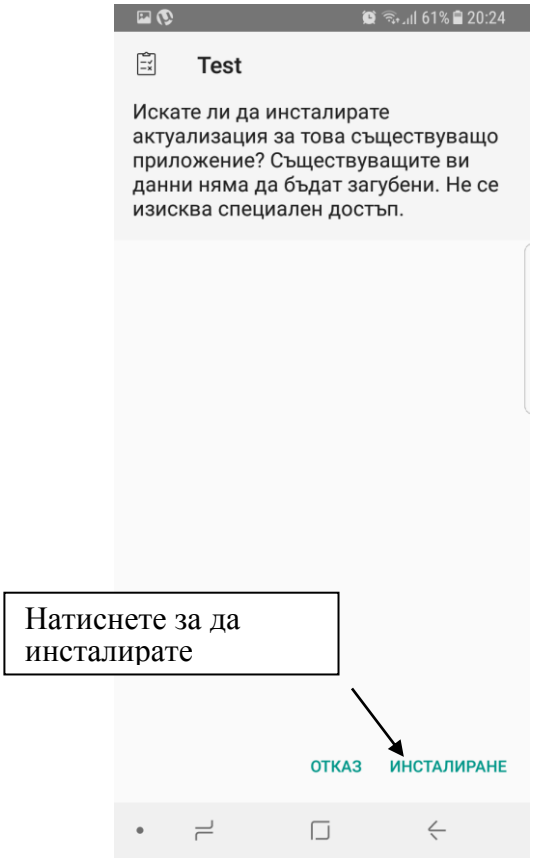
5.1 Инсталиране на приложението

За студентите вече щом имат apk файл единственото което трябва да направят е да натиснат върху него и да го инсталират на телефонните си устройства



След това просто чакат приложението да се инсталира и да бъде готово за използване на телефона.

По – долу ще бъдат предоставени изображения за инсталирането на приложението стъпка по стъпка за по – голямо изяснение и уточнение на студентите.



5.2 Въвеждане на факултетен номер, курс и година на обучение на студента и избиране на тест

В тази стъпка студентите ще трябва да въведат факултетният си номер, курса и годината, преди да си изберат тест за решаване.

Много е важно този факултетен номер да е правилен, защото той ще идентифицира студента и съответно ще го допусне до избор на тест за решаване.

Като казвам, че факултетният номер е много важен го казвам защото в java сървъра съответно за това клиент-сървър приложение е поместена цялата база данни с която тази версия работи.

За в бъдеще логиката ще бъде разширена с възможността този сървър да се предостави за работа със преподаватели

5.3 Решаване на съответния тест

В тази стъпка студентите ще трябва изберат отговор на избрания тест. Като в нашият вариянт се избира от три варианта

Question_table, което е всъщност таблица от базата данни за Android_test

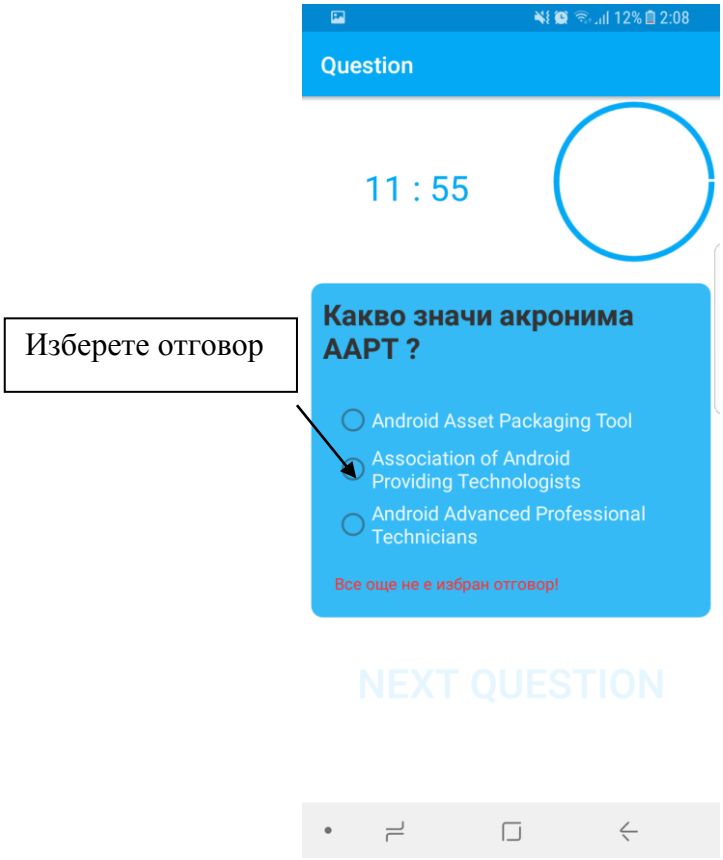
CSharpQuestion_table, което е таблица в базата данни за CSharp_test

JavaQuestion_table, което е таблица в базата данни за Java_test

Това се повтаря общо 6 пъти т.е. има 6 произволно генериран въпроса, който могат да бъдат всеки един от възможни 3 типа.

Първи тип. Или така наречения тип с един възможен отговор от три отговора. За този тип въпроси може студента да получи общо 5 точки.

Тук допълнително ще се поясни на студена и съответно ще се разгледат и 3те типа малко по-подробно,като за всеки един тип който е споменат по-горе ще се приложат снимки на екраните.

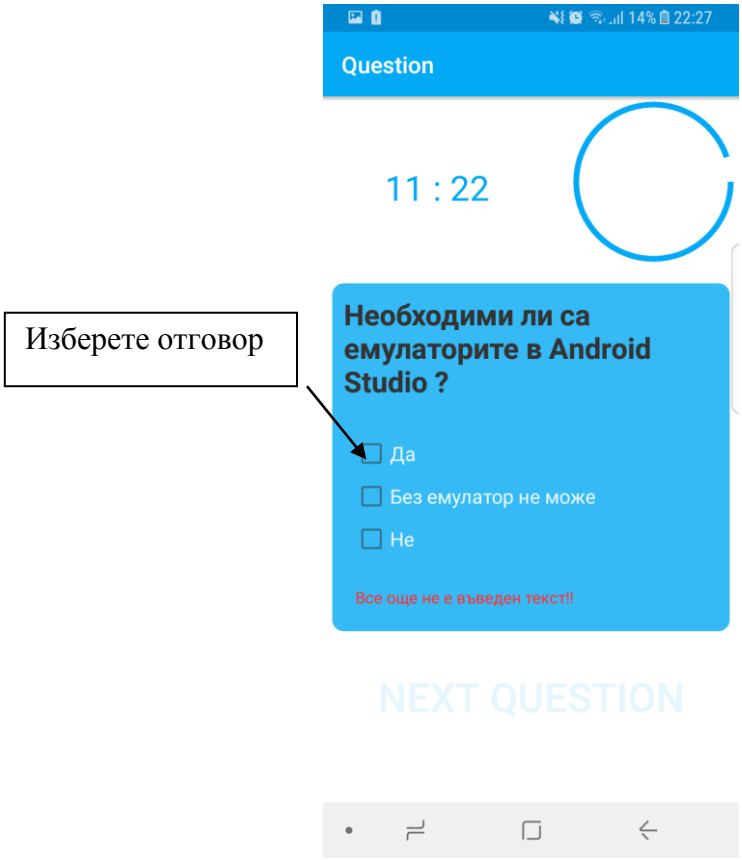


Първи тип. Или така познатият на всички студенти тип с един единствен правилен отговор от възможни съответно няколко отговора.

Тук студента трябва внимателно да обмисля, кой точно е съответният правилният отговор, за зададеният въпрос, защото от това зависият точките които ще получи.

Ако не знае кой е правилният отговор или даде грешен според него отговор тогава той няма да получи никакви точки за този въпрос.

Идята за точкуването е описано по-подробно горе в предишната страница на документацията.

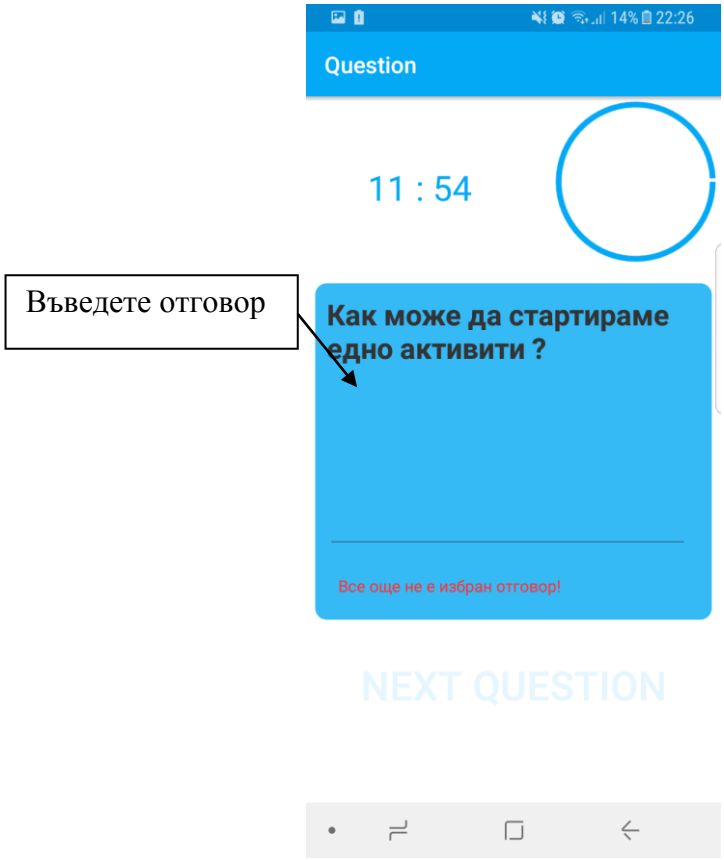


Втори тип. Или така наречения тип с множествени отговори от възможно три отговора за зададеният въпрос.

В тази ситуация дори да се избере само един отговор на студента ще му бъде разрешено да премине нататък към скедвашият генериран произволно въпрос, като идеята тук е студента сам да се сети, че има повече от един правилен отговор, разбира се той ще получи точки дори и на един правилен отговор, но те няма да са пълнятя брой, който се предостава за теста.

Тук в преимущество е стъдента защото, няма вариант в който тои да обърка всички отговори, поне един от верните отговори ще е познал и без значение колко малко пак му се дават точки за отговора.

Отново пояснявам, че за този тип отговори студента може да получи общо 5 точки, ако естествено е отговорил правилно на въпроса, ако не е отговорил правилно тогава ще му се дадът 0 точки от системата.



Трети тип. Или така наречения тип където студентите трябва да дадат свободен отговор и той трябва да бъде проверен по – късно от преподавателя.

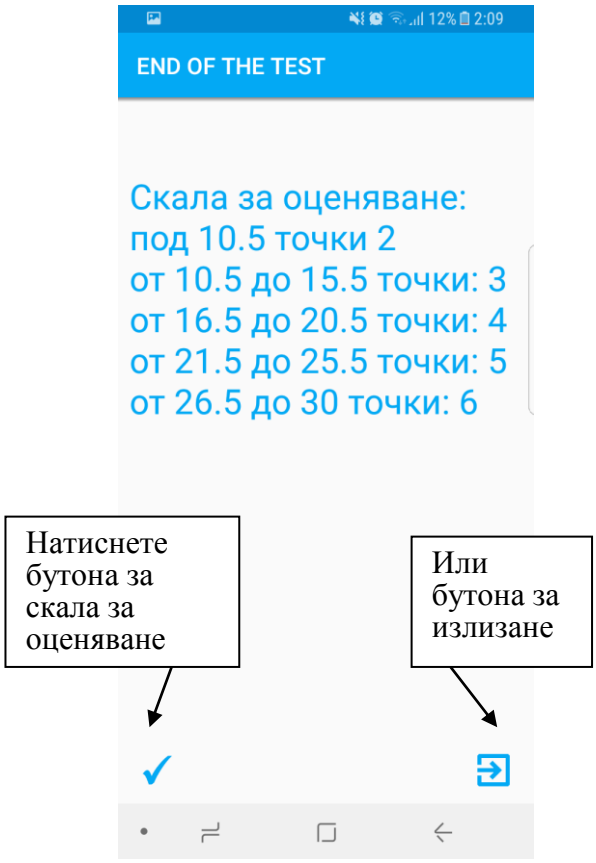
В тази ситуация се покза как разсъждава студента като той трябва да се изрази със свои собствени думи.

Тук трябва да се внимава, защото, изказа, който студента ползва трябва да е достатъчно ясен за преподавателя.

Ако се използва ваше обяснение на зададеният въпрос, а не дословно „рецитиране" на отговора тогава и този вариянт би бил приемлив от преподавателя.

Пак пояснявам, че ако е правилен отговора отново ще се дадат за него на студента общо 5 точки, ако на преподавателят не му стане достатчно ясно за какво говори студента или има не достатъчно ясен изказ, както горе бе споменато тогава точни няма да му се присъдят за даденият отговор.

5.4 Проверяване на крайна скала за оценяване и получаване на оценка



Тук на базата на скалата за оценяване, която може да се изкара при натискането на бутона с тикчето, намиращо се най долу в ляво студентите могат да преценят на каква оценка са изкарали за теста и колко добре са се представили и дали ще могат да се отличат в бъдещата ранкинг система, която ще се направи за предстоящи версии на приложението. Съответно копчето, което се намира в дясно е за да се затвори приложението след края на теста.

6 Заключение

Благодарение на архитектурата на цялосното приложение (както сървъра така и клиента) и бъдещо дообработване, то може да стане, както в началото споменах, достъпно дори и за детската градина.

Също така клиент-сървър приложението, с тази архитектура, е отворено за добавяне на нови пакети и съответно и нови класове, за разширяване на логиката му.

При подобряване на програмата чрез добавяне на нови възможности, те могат да се реализират без да е нужно преработване на големи части от програмния код, а само чрез добавяне на нов код. Този факт прави подходяща програма за развитие в бъдеще.

Приложението е така направено, че до този момент да е в предимство на студента, защото не му се налага да е в самият университет за да реши тест, а просто си пуска приложението и избира тест, като разбира се това става с помощта на:

```
<uses-permission android:name="android.permission.INTERNET" />
```

user permission-а в андроид приложението.

За финал, бих казал, че целта е постигната (разработването на клиент - сървър приложение за тестване на знанията на студентите)

За бъдещо разработване се очаква идеята за разширяване на самото приложение, така,че да е възможно ползване не само в университетски среди (висше образование), но и в други среди, като началното образование, средното образование и да остане удобно за ползване от студенти, но да е удобно и за преподаватели, като частта със сървъра, който за сега ползва готови таблици с бази данни, да е предоставена за употреба от учи

7 Използвана литература

Документът е предназначен за четене от членовете на комисията, тестери, и съветници. Предложена литература за четене:

7.1 Английска литература:

THE Java™ Programming Language, Fourth Edition - Ken Arnold ,James Gosling, David Holmes

Android 4 platform SDK techniques for developing smartphone and tablet applications - Satya Komatineni, Dave MacLean

7.2 Българска литература:

Въведение в програмирането с JAVA. - Светлин Наков и колектив

7.3 Помощни страници и линкове:

Google.bg

Stackoverflow.com

линк:

https://stackoverflow.com/questions/27213381/how-to-create-circular- progressbar-in-android

за създаването на curcular progress bar

линк:

https://stackoverflow.com/questions/5197892/add-shadow-to-custom-shape-on-android

за създаването на drawable shadow

линк:

https://stackoverflow.com/questions/11288475/custom-toast-in-android-a-simple-example за създаването на custom тост за приложението

линк :

https://stackoverflow.com/questions/26311785/android-drawing-custom-shapes

за създаването на custom формички

линк:

https://stackoverflow.com/questions/4908114/how-to-use-onpause-with-android

как да се работи с onPouse() специално за Timer-а на приложението

линк:

https://stackoverflow.com/questions/15658687/how-to-use-onresume

как да се работи с onResume() отново специално за Timer-а на приложението

линк:

https://stackoverflow.com/questions/18002227/why-extend-an-application-class

как да се използва Application class

линк:

https://stackoverflow.com/questions/2929562/register-application-class-in-manifest

къде да се регистрира използването на Application class

Developer.android.com

с линк:

https://developer.android.com/reference/android/os/AsyncTask.html

за ползването на AsyncTack в android

линк:

https://developer.android.com/guide/topics/resources/drawable-resource.html

drowable shapes tutprial

линк: https://developer.android.com/design/material/index.html

повече за Material design

Javacodegeeks.com

Mkyong.com

линк:

https://www.mkyong.com/java/how-do-convert-java-object-to-from-json-format-gson-api/

за ползването на gson с java.

линк:

https://www.mkyong.com/servlet/a-simple-servlet-example-write-deploy-run/

За използването на java servlet-и.

Codezheaven.com

линк:

http://www.coderzheaven.com/2017/01/19/using-gson-in-android-simple-demo/

за ползването на gson с android

Tutorialspoint.com

линк:

https://www.tutorialspoint.com/servlets/servlets-first-example.htm

за ползването на Servlet-и в java.

Vogella.com

линк:

http://www.vogella.com/tutorials/AndroidDrawables/article.html

какво са drawable ресурси и как се ползват

линк:

http://www.vogella.com/tutorials/AndroidBackgroundProcessing/article.html

за AsyncTask Example