**ТЕХНОЛОГИЧНО УЧИЛИЩЕ ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

**ДИПЛОМНА РАБОТА**

Тема: Traders

(Платформа за търговия на стоки)

Дипломант: Научен ръководител:

*Валентин Върбанов Александър Осенов*

СОФИЯ

2018

**TЕХНОЛОГИЧНО УЧИЛИЩЕ**



**ЕЛЕКТРОННИ СИСТЕМИ**

**към ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - СОФИЯ**

Дата на заданието: 06.11.2017 г. Утвърждавам:..............................

Дата на предаване: 06.02.2018 г. /проф. д-р инж. Т. Василева/

**ЗАДАНИЕ**

**за дипломна работа**

на ученика Валентин Викторов Върбанов 12 Б клас

1.Тема: Платформа за търговия на стоки

2.Изисквания:

- Да се създаде УЕБ платформа, позвлояваща търговия на стоки.

- Android приложение улесняващо използването на платформата както от търговци, така и от масови потребители.

- Android приложението трябва да автоматизира процесите за поръчки,

менажиране на стоки и управление на лоялни клиенти.

3.Съдържание 3.1 Обзор

3.2 Същинска част

3.3 Приложение

Дипломант :...........................................

Ръководител:..........................................

/Aлександър Осенов/

Директор:................................................

/ доц. д-р инж. Ст. Стефанова /

# УВОД

Развитието на по-малките фирми и дори самостоятелни търговци не е едно от най-лесните неща за постигане, но в днешни времена с всеки следващ изминал ден се появяват нови и нови приложения, някои от които подпомагат именно това.

В последното десетилетие са създадени множество приложения, целящи да предоставят и улеснят достъпа на тези по-малки фирми и самостоятелни търговци до пазара. Тези приложения работят чрез обяви, които търговецът „поства“, след което всеки заинтересован може да преглежда тези обяви, търсейки нещо специфично или изгодна сделка.

Целта на настоящата дипломна работа е да се разработи система, която да прави точно това, а именно да помага на непознати и малки фирми и търговци да достигнат до своите клиенти, помагайки им да се развиват.

# ГЛАВА I

## 1.1 Приложения за търговия на стоки

Съществуват няколко приложения за търговия на стоки. Някои от тях имат по голям избор от категории, а други позволяват по-лесен достъп до локацията на продавача, давайки идея за това, колко лесно да бъде приемането на купената обява.

В рамките на проучването бяха разгледани две приложения: OLX и bazar.bg.

## 1.1.1 OLX

Фиг.1.1

**Прилики:**

Мобилното приложение OLX, също както и разработеното за дипломна работа приложение, позволява:

1.) Регистрация на акаунт

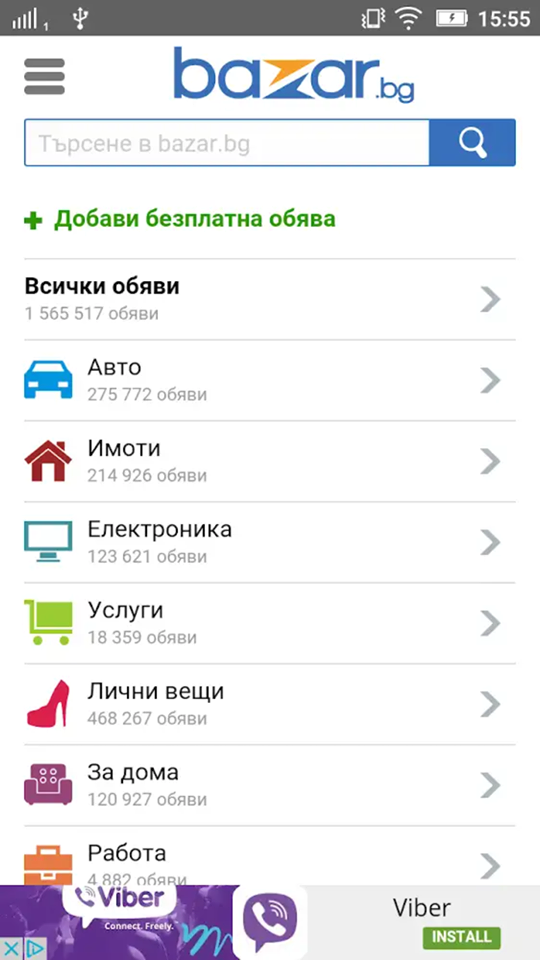
2.) Богат набор от категории за отделните видове обяви

**Разлики:**

Създаденото приложение е по-строго разделено на продавачи и купувачи. Това го прави по-професионално настроено и по-лесно за навигация, предоставяйки на потребителя по малко опции, които да могат да го объркат.

Създаването на обяви е достъпно само за акаунтите на търговци, а купувачите могат да разглеждат всички тези обяви, както и всеки един търговец.

## 1.1.2 Bazar.bg



Фиг.1.2

**Прилики:**

Мобилното приложение bazar.bg, също както и разработеното за дипломна работа приложение, позволява:

1.) Регистрация на акаунт

2.) Богат набор от категории за отделните видове обяви

**Разлики:**

Създаденото приложение е по-строго разделено на продавачи и купувачи. Това го прави по-професионално настроено и по-лесно за навигация, предоставяйки на потребителя по малко опции, които да могат да го объркат.

Създаването на обяви е достъпно само за акаунтите на търговци, а купувачите могат да разглеждат всички тези обяви, както и всеки един търговец. Има възможност и за използване на QR кодове.

## 1.2 Основни среди за разработка на Android приложения

## 1.2.1 Eclipse IDE

Eclipse е среда за разработване на софтуер, която се състои от IDE (интегрирана среда за разработка) и от plugin система. Тази среда е написана основно на Java и също така е най-често използвана за Java приложения.  
Еclipse предоставя на потребителите си възможността да инсталират или да създадат свои плъгини и така да разширят функционалността на средата. Всеки плъгин се интегрира в Eclipse средата по един и същ начин. Всичко в Eclipse, с малки изключения, са плъгини. Част от тях предоставят на потребителя да разработва и приложения на други езици.

Eclipse е безплатен и с отворен код. Появява се през 2001 година, а от 2004 за разработката и поддръжката му се грижи Eclipse Foundation – организация, част от която са част от най-големите технологични компании (IBM, Oracle и т.н).

ADT (Android Development Tools) плъгина на Eclipse осигурява среда за разработка и на Android приложения. Този плъгин помага на разработчиците да създават и разпространяват своите Android проекти, да ги дебъгват и да създават .apk (android application package) файлове.

Плъгинът е безплатен и е официална среда за разработка на този тип приложения до създаването на Android Studio (базиран на  IntelliJ IDEA).

От 2016 година Google спират поддръжката на ADT, за да може повече разработчици да преминат на вече официалната среда за разработка -Android Studio. Въпреки това Eclipse все още предоставя възможността за създаване на Android приложения, но липсата на поддръжка затруднява разработването им.

## 1.2.2 Android Studio

Android Studio е официалната среда за разработка на Android приложения. Версия 1.0 се появява през декември 2014г. и се поддържа от Google и JetBrains. Базира се на JetBrains’ IntelliJ IDEA и е създадена специално за разработването на Android, като така заменя Eclipse ADT.  
Android Studio предоставя голям набор от най-добрите и бързи инструменти за създаване на приложения за всякакъв тип Android устройства.   
Лесен е за ползване и се разработва и подобрява активно. Безплатен е за ползване. Използва Gradle и предоставя възможност за работа с емулатори, които дават възможност бързо да се провери как точно ще изглежда и действа разработваното приложение на конкретно устройство при зададен размер и резолюция на екрана. Средата позволява удобно компилиране на .apk файлове с различна функционалност от един проект, така че лесно могат да се създават и няколко версии на приложението с различни възможности. Освен това в IDE средата има и готови “чернови” за различни видове приложения и интеграция с GitHub.

Средата за разработка Android Studio е достъпна във версии за Windows, Mac и Linux.

## 1.2.3 B4A

B4A – Basic4Android (Basic for Android) е среда за разработка на приложения, позволяваща да се направят максимално леки, бързи и лесни за разработка приложения. B4A е алтернатива на Android Studio и Java, базирано на Visual Basic, но е адаптиран за андройд средата. Програмата също така включва много удобен и практичен дизайнер за приложението, улесняващ работата на програмиста. Най-голямото предимство е улеснената работа с различни размери телефони и таблети, тъй като B4A може да оразмерява това, което клиента вижда автоматично.

# ГЛАВА II

## 2.1 Изисквания към дипломната работа

Проектът се състои от две главни части:

1. Сървър
2. Мобилно приложение

## 2.1.1 Traders Mobile

- Позволява създаването на два вида акаунти:

- Продавач

- Купувач

- Купувачът има право да разглежда всички публикувани от търговците обяви, след което му се предоставя възможност да се свърже с търговец по избор.

- Продавачът има право да създава нови обяви, като избере категорията на дадения артикул и го опише накратко. Приложението му показва когато даден потребител е заинтересован от негова обява и му предоставя възможност да се свърже с потенциален купувач.

- Могат да се качват снимки на артикула за визуализация.

- Има QR код, който помага на купувачите да намерят любими и често използвани стоки или любимите си търговци.

## 

## 2.1.2 Traders Server

- Цялата база данни се достъпва чрез сървъра. Тя се състои от множество таблици, в които се запазва информацията за всички функции на мобилното приложение, акаунтите на потребителите и техните обяви.

- Предоставя множество ендпойнти, отговарящи за триенето, създаването, коригирането на обяви, както и за регистрацията на нови потребители.

## 2.2 Избор на език за програмиране

За език за програмиране беше избран Java, който е обектно-ориентиран език за програмиране появил се през 1995г. За да се изпълни дадена програма, написана на Java, е необходима виртуална машина JVM (Java Virtual Machine). Това се налага поради факта, че Java кодът не се компилира до машинен код, а до т.н. байт код – специфичен за езика код. Тази реализация има много предимства, които значително надвишават недостатъците.   
Някой от предимствата на Java са лесната ѝ преносимост между различните платформи, защото веднъж компилирана, една програма може да бъде стартирана на всяко устройство с виртуална машина, независимо от архитектурата или операционната му система. Тази виртуална машина осигурява и други предимства като garbage collector (освобождаване на паметта от обекти, които вече не се използват и към тях няма повече референции), heap (предварително заделяне на динамична памет), възможност за контрол на правата на потребителя на ниво виртуална машина, висока степен на сигурност (няма директен достъп до паметта) и т.н.   
Основният недостатък на езика е необходимостта от допълнителни ресурси, като процесорно време и памет, за изпълнението на самата виртуална машина, както и по-ниската производителност.  
Java е език от високо ниво и всичко в него се дефинира в класове, които са основна концепция на езика. Класовете имат свойства и методи (определящи поведението на един клас). Тук не съществуват глобални променливи.

Всички тези предимства на езика, както и голямата общност, използваща го, прави Java едно от най-добрите решения за разработването на приложения.

## 2.3 Избор на среди за разработка

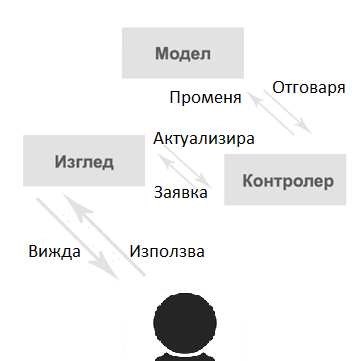
## 2.3.1 Избор на среда за разработка на Android приложението

От 2015г. Android Studio e официалната среда за разработка на Android приложения и е поддържана от Google. Освен, че е безплатна и лесна за ползване, тя се разработва и подобрява активно и предоставя много полезни функционалности като емулатори за всякакъв вид устройства, използващи Android операционна система, базови проекти с различни функционалности (google maps, admob и т.н.) и др. Android Studio е достъпен за Windows, Mac и Linux.   
Бързото развитие на Android Studio в последните години го прави най-доброто решение за създаването на Android приложения за всякакъв вид устройства.

## 2.3.2 Избор на среда за разработка на сървъра

Spring Framework е технологична рамка с отворен код, която предоставя множество функции и класове, улесняващи разработването на Java EE (Enterpise Edition) приложения. Появява се през 2002г. и включва различни модули, предоставящи голям брой различни функционалности, като управление на транзакциите, аспектно-ориентирано програмиране, контейнер на зависимостите, достъп до данни и др.

## 2.4 Избор на архитектура на приложението



Фиг.2.1

За архитектура на приложенията бе предпочетена MVC архитектурата – модел-изглед-контролер. Той е основан на разделянето на бизнес логиката от данните и графичния интерфейс на приложението. Съставен е от три части:

* Модел – това са данните, с които работи приложението. Тези данни се въвеждат, променят, визуализират и т.н. Моделът се манипулира посредством контролера, а промяната в данните му води и до промяна на изгледа , подавайки отговор на контрлера.
* Изглед – отговорен е за визуализацията на модела и представянето му на потребителя, по лесен за разбиране и използване начин.
* Контролер –работи с данните от модела, като ги преработва и ги подава в подходящ вид на изгледа. Отговорен е за връщането на данните от модела към изгледа, позволявайки на потрябителя да види резултата от направената промяна или заявка.

Предимствата на MVC са, че моделът е независим от останалите части на системата и може да бъде планиран и осъществяван независимо от тях. За един и същи модел могат да бъдат създавани различни изгледи. Контролерът и изгледът също могат да бъдат променяни, без това да налага промяна и в модела.

## 2.5 Избор на база данни

Базата данни на приложението е създадена на MySQL, която е релационна база данни, структурирана в отделни таблици. При големи бази или при бази с много таблици, MySQL има предимство пред нерелационните бази данни, поради факта, че по този начин се постига по голямо разделение на отделните части, правейки редакцията, добавянето и триенето от базата много по-лесно.

MySQL е една от най-лесните за изучаване бази данни и притежава голяма общност от разработчици, а също така за нея може да се намери голям обем от информация в интернет.

В базата се записват не само регистрираните потребители и техния тип, но също така и всички обяви, отделните предмети и обекти съставляващи ги, както и склада и наличните в него стоки на всеки търговец.

## 2.5.1 Структура на база данни – артикул

* Id

Уникален идентификатор, който се създава автоматично при добавяне на нов артикул в базата и помага информацията за него да се взима по-лесно.

* Name

Името на даден артикул.

* basePricePerUnit

Базовата цена на даден артикул. Избира се от търговеца. По-късно се слагат надценки на артикула над тази цена, с цел печалба.

* Description

Описанието на даден артикул.

* Categories

Категориите, в които участва даден артикул. Един артикул може да принадлежи към повече от една категория.

* Unit

Мярката, с която се измерва артикулът(кг, бр и тн.).

* object

Обект за даден артикул, съдържащ свойствата му(цвят, размер и др.).

## 2.5.2 Структура на база данни – Категория

* id

Уникален идентификатор, който се създава автоматично при добавяне на нова категория в базата и помага информацията за нея да се взима по-лесно.

* categoryName

Името на дадена категория

* categoryLevel

Ниво на дадена категория. Всяка подкатегория има различна стойност на тази променлива.

* items

Артикулите, които отговарят на тази категория.

## 2.5.3 Структура на база данни – Мерна Единица

* id

Уникален идентификатор, който се създава автоматично при добавяне на нова мерна единица в базата и помага информацията за нея да се взима по-лесно.

* name

Името на мерната единица(кг, бр., л и т.н.)

## 2.5.4 Структура на база данни – Потребител

* id

Уникален идентификатор, който се създава автоматично при добавяне на нова категория в базата и помага информацията за нея да се взима по-лесно.

* username

Името на даден потребител.

* password

Паролата на даден потребител.

* email

Имейлът, използван за регистрация на даден потребител.

* phone

Телефонният номер на даден потребител, ползван за предоставяне на връзка с потенциални купувачи.

* accountType

Видът на акаунта. Има два вида акаунти:

* Купувач
* Търговец/Продавач

## 2.5.5 Структура на база данни – Склад

* id

Уникален идентификатор, който се създава автоматично при добавяне на нова категория в базата и помага информацията за нея да се взима по-лесно.

* quantity

Броят на наличните артикули от обявата.

* priceMultiplier

Надценката над базовата цена на артикула. Избира се в проценти и това представлява печалбата на търговеца над базовата цена. Това поле е невидимо в обявата.

* qRHash

Стринг от символи, използван за създаването на QR код чрез библиотека в андроид приложението.

* dealer

Връзка към търговеца, публикувал дадена обява.

* item

Връзка към даден артикул от обявата.

* image

Снимка на артикула, качена в обявата от търговеца. Тя се запазва в тип BLOB(Binary Large Object) и се ползва за по-лесното и олекотено запазване на снимката.

## 2.5.6 Структура на база данни – Обект

* id

Уникален идентификатор, който се създава автоматично при добавяне на нова категория в базата и помага информацията за нея да се взима по-лесно.

* objectName

Името на обекта (кола, стол, маса), идентифициращо какви свойства има дадения артикул.

* properties

Лист връзка към свойствата на артикула, в зависимост от името (вида) на артикула.

## 2.5.7 Структура на база данни – Свойства

* id

Уникален идентификатор, който се създава автоматично при добавяне на нова категория в базата и помага информацията за нея да се взима по-лесно. Създава се обща таблица от свойства и обекти.

* propertyName

Името на дадено свойство (цвят).

* value

Стойността на дадено свойство (червен).

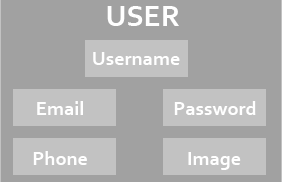
* myObjects

Връзка към лист от обекти, съдържащи дадено свойство.

# ГЛАВА III

## 3.1 Основни модели

## 3.1.1 Модел Потребител

****

Фиг 3.1

Моделът потребител е един от по-простите модели в приложението. Той съдържа данни за потребителя:

* Име
* Имейл
* Парола
* Телефонен номер
* Вид потребител

Използва се най-често при регистрация и влизане в акаунт, но също така има участие и в обявата (при търговски акаунт) и профила.

## 3.1.2 Модел Свойства



Фиг 3.2

Моделът Свойства съдържа две полета:

- Името на свойството

- Стойността на свойството

Този модел се използва, за да се добавят различните свойства на различните артикули. Например: една кола ще има марка, цвят, конски сили и т.н., докато масата ще има материал, височина, цвят. Свързвайки се с модела, за един обект става лесно разпределянето на тези свойства в артикулите.

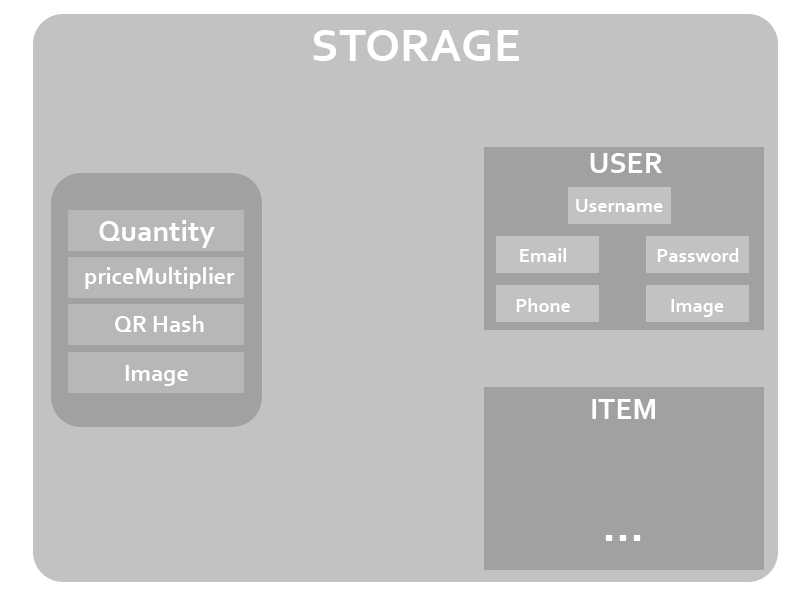
## 3.1.3 Модел Обекти

****

Фиг 3.3

В модела се съдържат име на вида и артикул. Той се свърза с модела за свойства, след което се добавя към всеки артикул, добавяйки разнообразие на свойствата в зависимост от вида на артикула (мебел, храна, техника и тн.).

## 3.1.4 Модел Склад



Фиг 3.4

Този модел е един от главните модели на приложението и се използва основно за обявите. При създаване на нова обява, се попълва моделът Склад и през него се попълва отделният артикул, категорията му и неговите свойства. Създава се QR код, който да прехвърля към тази страница. Той съдържа няколко полета:

* Бройката на артикулите в обявата.
* Надценката над базовата цена на артикула.
* Информация за търговеца, публикувал обявата.
* Информация за артикула, публикуван от търговеца.
* Стринг, използван за създаването на QR код в приложението.
* Снимката, запазена в обявата.

## 3.1.5 Модел Мерна единица

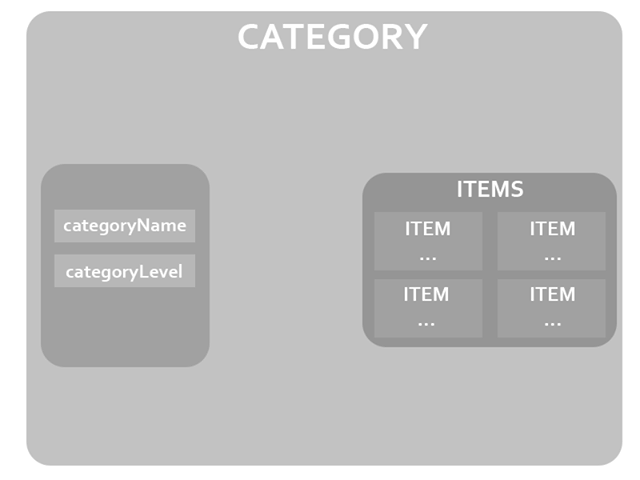


Фиг 3.5

Този модел е сравнително малък. Той позволява да се съхрани мерната единица на различните видове артикули. Някои артикули се измерват в литри, а други - в килограми. Този модел запазва тази информация и притежава само две полета:

* Име на мерната единица (напр. килограм).
* Съкращението на тази мерна единица (напр. кг).

## 3.1.6 Модел Категория

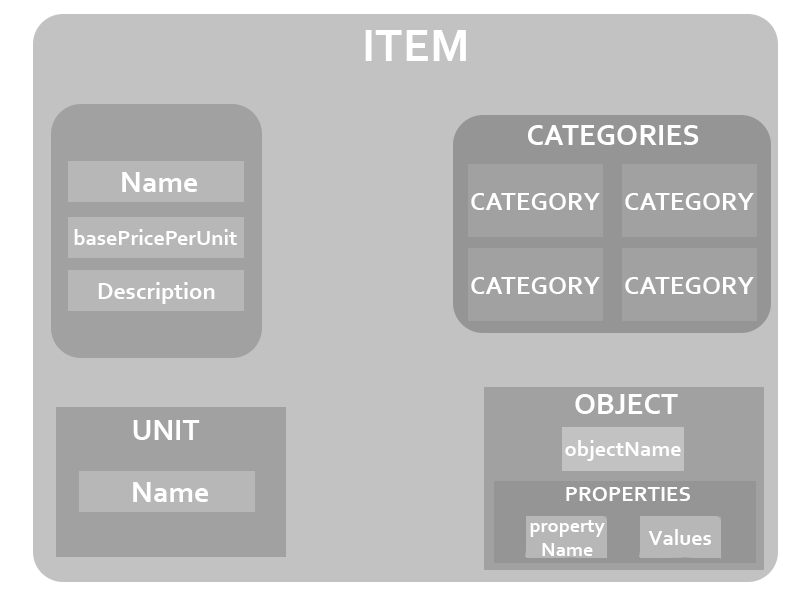


Фиг 3.6

Това е друг важен модел за приложението. Той отговаря за правилното подреждане и разделение на различните артикули. Има две нива на категориите, като първите са главните категории, а другите са подкатегории. Всяка подкатегория има референция към бутона, избран да зареди правилната подкатегория. В този модел има няколко параметъра:

* Името на категорията
* Артикулите, спадащи към тази категория
* Променлива, показваща дали записът е категория или подкатегория.

## 3.1.7 Модел Артикул



Фиг 3.7

Това е главният модел в приложението, съдържащ цялата информация за всеки отделен артикул. Той се попълва при създаването на нова обява и след това се взима, когато даден потребител достъпи неговата категория. Има следните параметри:

* Име на артикула.
* Базова цена, използвана за референция преди да се добавят надценка и ДДС.
* Описание на артикула, въведено от търговеца.
* Категориите, към които спада артикулът.
* Мерната единица на артикула.
* Какъв обект е артикулът и свойствата му.

## 3.2 Създаване на обява

Цялата идея на приложението е базирана на обяви. Обявата се създава от менюто за нови обяви на андроид приложението. След попълване на зададените полета и завършване на обявата, полетата се валидират и след това се изпращат на сървъра под формата на DTO(data transfer object), посредством Retrofit заявка.

При успешно създаване на обявата тя се записва в няколко модела и се запазва в базата.

## 3.2.1 Retrofit

Ретрофит е REST клиент за Android. Той служи за опростеното и бързо изпращане на данни по множество различни начини (JSON,XML и т.н.). Избира се какъв вид данни ще се подават и приемат и какъв конвертор ще се използва. Използва се OkHttp и са нужни 3 класа за пълноценната работа с Retrofit:

* Модел
* Интерфейс, дефиниращ HTTP операциите
* Retrofit.Builer, който отговаря за създаването на ендпойнти, управляващи извършването на заявките.

dependencies {

// Retrofit & OkHttp

compile 'com.squareup.retrofit:retrofit:1.9.0'

compile 'com.squareup.okhttp:okhttp:2.7.2'}

Фиг.3.8

Зависимост (dependency), нужна за интеграция на Retrofit и OkHttp в gradle.

String API\_BASE\_URL = "https://api.github.com/";

OkHttpClient.Builder httpClient = new OkHttpClient.Builder();

Retrofit.Builder builder =

new Retrofit.Builder()

.baseUrl(API\_BASE\_URL)

.addConverterFactory(

GsonConverterFactory.create()

);

Retrofit retrofit =

builder

.client(

httpClient.build()

).build();

GitHubClient client = retrofit.create(GitHubClient.class);

Фиг.3.9

Създава се retrofit клас, който е нужен за всички заявки. След това се създава клиент, който отговаря за изпълнението на самата заявка.

## 3.3 Зареждане на данните в мобилното приложение от базата данни

След като търговеца е създал обява, тя може да бъде заредена от други потребители в мобилното приложение. Когато потребителят избере дадена категория, всички обяви, включващи артикули, които отговарят на избраната категория ще се покажат на екрана, под формата на лист. След това потребителя може да преглежда всяка от обявите и да се свърже с търговеца. За заявката отново се използва Retrofit, който получава информация, нужна за визуалното представяне на обявите. За по-бързо и ефикасно зареждане на картините, включени в обявите се използва библиотеката Glide.

## 3.3.1 Glide

Glide е бърза и ефикасна рамка с отворен код, разработена за андроид, която специализира в менажирането на различни медия файлове и тяхното декодиране или кеширане. Поддържа визуализирането на снимки, видеа и дори GIF файлове. Представлява лесно за включване в проекта API, което ползва персонализиран HttpUrlConnection, но може да се зададе използването на OkHttp или други google стакове при предпочитание. Главната цел на библиотеката е да направи прегледа и движението на какъвто и да е лист от снимки или видеа колкото се може по лесно, но също така помага и при вземане, визуализиране и преоразмеряване на снимки или друг вид медия.

dependencies {

 implementation 'com.github.bumptech.glide:glide:4.6.1'

annotationProcessor 'com.github.bumptech.glide:compiler:4.6.1'

}

Фиг.3.10

Библиотеката има голям набор от прости, но много полезни методи. Като например метод за зареждане на проста снимка:

// For a simple view:

@Override public void onCreate(Bundle savedInstanceState) {

...

ImageView imageView = (ImageView) findViewById(R.id.my\_image\_view);

GlideApp.with(this).load("http://goo.gl/gEgYUd").into(imageView);

}

Фиг.3.11

Както и за лист от снимки:

// For a simple image list:

@Override public View getView(int position, View recycled, ViewGroup container) {

final ImageView myImageView;

if (recycled == null) {

myImageView = (ImageView) inflater.inflate(R.layout.my\_image\_view, container, false);

} else {

myImageView = (ImageView) recycled;

}

String url = myUrls.get(position);

GlideApp

.with(myFragment)

.load(url)

.centerCrop()

.placeholder(R.drawable.loading\_spinner)

.into(myImageView);

return myImageView;

}

Фиг.3.12

За версия 4 на библиотеката е нужна SDK версия на API 26 или нагоре, като минималното е API 14.

## 3.4 Защита JWT

За защита на приложението бе избран JWT (JSON Web token). Сървърът създава токен, който притежава някакви права и след това го подава на потребителя. Ако сървърът подаде на даден потребител токен, който показва, че получилият го потребител е администратор, то тогава този токен се използва за доказателство. Сървърът подписва токена със своя ключ, правейки го валиден преди изпращане. Токените са създадени с цел сигурност, но и също така компактност. JWT използва JSON стандарти. Той се състои от 3 отделни части:

* header, който съдържа вида на криптиране(RS ключове или SHA256)

"alg": "HS256", "typ": "JWT"

Фиг 3.13

* payload, който съдържа името, типа акаунт(администратор или обикновен потребител) и променливау показваща кога е подаден токена на този потребител.

"sub": "1234567890", "name": "John Doe", "admin": true

Фиг 3.14

* Подпис на токена, състоящ се от ключ, зададен в сървъра, header и payload (събрани в един стринг и разделени с точка) и самият подпис, състоящ се от хеширания стрингq създаден по-рано и ключаq зададен предварително. Този хеш също се добавя към стринга и се разделя с точка.

key = 'secretkey'

unsignedToken = encodeBase64Url(header) + '.' + encodeBase64Url(payload)

signature = HMAC-SHA256(key, unsignedToken)

Фиг.3.15

token = encodeBase64Url(header) + '.' + encodeBase64Url(payload) + '.' + encodeBase64Url(signature) # token is now: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJsb2dnZWRJbkFzIjoiYWRtaW4iLCJpYXQiOjE0MjI3Nzk2Mzh9.gzSraSYS8EXBxLN\_oWnFSRgCzcmJmMjLiuyu5CSpyHI

Фиг.3.16 Пример за краен токен

## 3.5 Изтриване на обява

Изтриването става много просто. При натискане на обявата, търговеца може да избере да я премахне, при което андроид приложението изпраща заявка през Retrofit, съдържаща информацията на обявата. Сървърът обработва заявката и намира артикула, включен в обявата и го изтрива, след което премахва обявата от листа с обяви.

## 3.6 Промяна на обява

Приложението поддържа и функция за редактиране на вече завършена обява, чрез бутона за промяна. Търговецът може да избере да промени обявата, при което се изпраща заявка към сървъра, изискваща информацията на избраната обява, след което потребителят се прехвърля към страницата за попълване на обява. Полетата там са попълнени с информацията на вече съществуващата обява. При завършване на промяната всички полета се запазват и заменят стария запис в базата чрез нова POST заявка.

# ГЛАВА IV

## Ръководство на потребителя

## 4.1 Traders

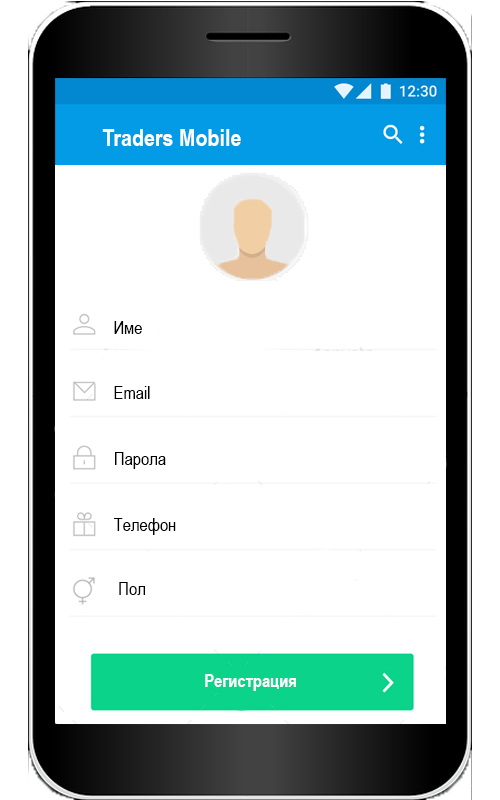
Системата е направена от две глвани части: Мобилно приложение и сървър. Първото предоставя начин на ползване за потребителите, като ги разделя на два вида: Продавачи(Търговци) и обикновенни потребители(Купувачи).

## 4.2 Мобилно приложение Traders

## 4.2.1 Инсталация

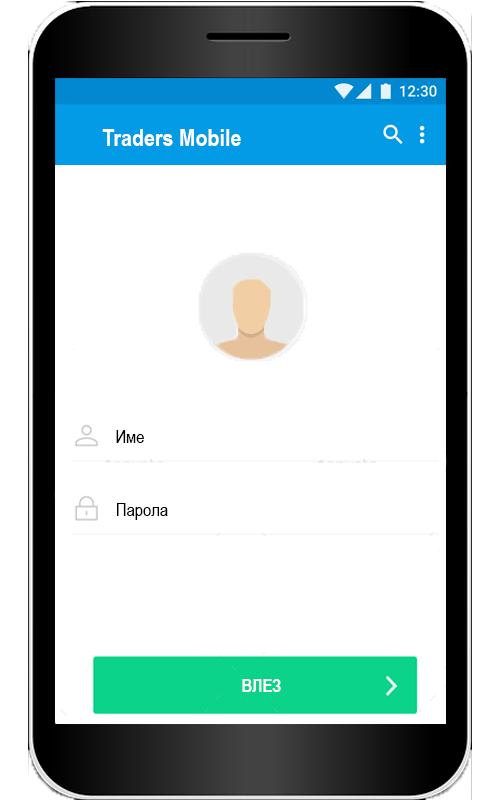
За инсталацията на мобилното приложение се отваря .apk файлът и се избира опцията за инсталиране. След успешна инсталация приложението може да бъде намерено при другите вече инсталирани приложения, откъдето то може да бъде стартирано.

**4.2.2 Регистрация**

****

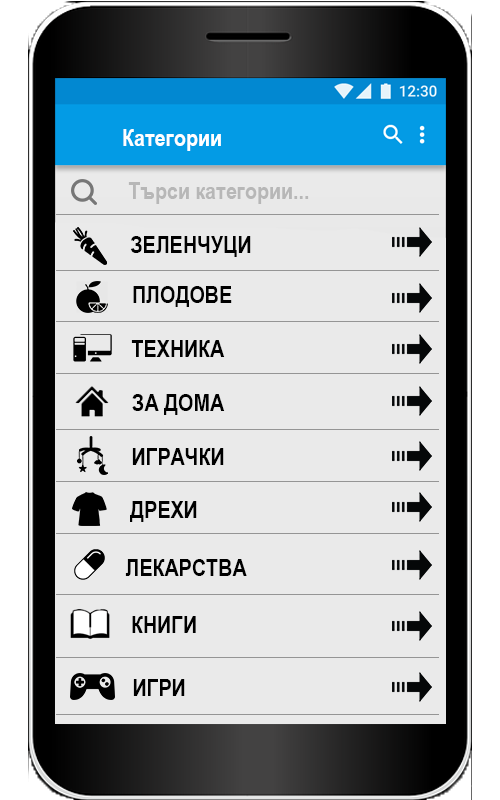
При първо стартиране на приложението се показва екранът за регистрация на потребител. Екранът се състои от няколко полета за попълване и един бутон „Регистрация“. При натискане на бутона полетата се валидират и ако регистрацията е успешна потребителят влиза в приложението и неговият акаунт се записва в базата данни.

## 4.2.3 Вход

****

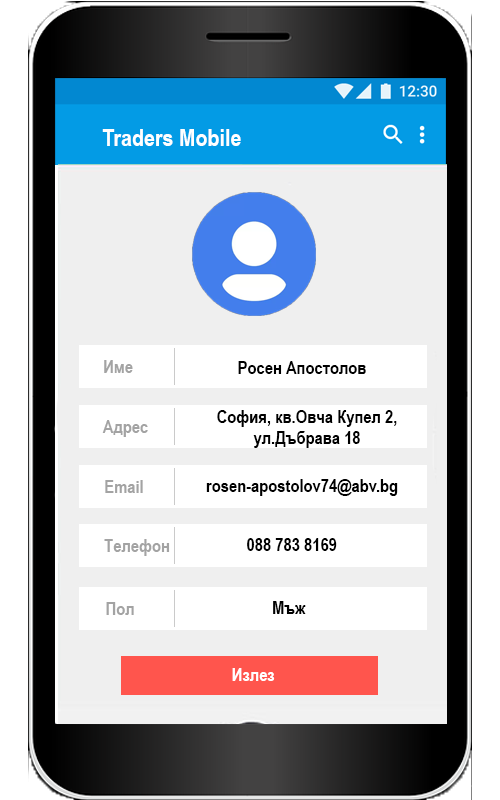
Когато приложението се стартира и сесията е изтекла или потребителят е излязъл от акаунта си ръчно се показва екран за влизане в приложението. Той е подобен на този за регистрация но използва единствено име и парола, зададени при регистрацията. При натискане на бутона „Влез“ полетата се появяват за съществуващ запис в базата данни на дадения акаунт и потребителят влиза в приложението.

## 4.2.4 Категории



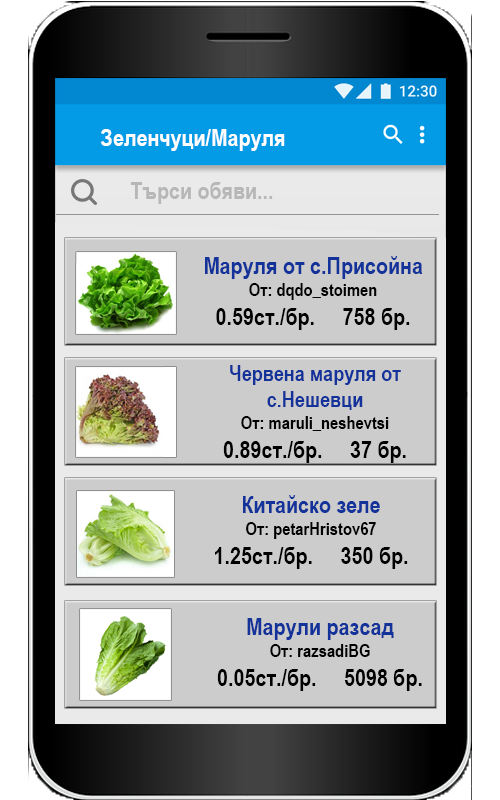
Чрез екранът с категории потребителят може да филтрира и намери артикули, които търси значително по-лесно. При избиране на категория екранът се сменя като сега показва подкатегориите на избраната категория. Потребителят може да избере подкатегория, което ще покаже всички обяви отговарящи на тази категория.

## 4.2.5 Профил



Профилът на всеки Търговец е видим за всички потребители. Този екран се състой от няколко полета съдържащи информация за избрания профил: Име, Приблизителен адрес, имейл за връзка, телефон за връзка, пол.

## 4.2.6 Обяви

****

След избиране на категория и подкатегория(Зеленчуци/Маруля) всички обяви отговарящи на тези категории се показват на потребителя. Може да се избере дадена обява, което отваря нов екран, съдържащ по-подробна информация за избраната обява, както и връзка към профила на търговеца създал тази обява, където може да се намери информация за него(Тел. номер, имейл).

За да се създаде обява, търговецът избира бутона за добавянен а нови обяви, след което се появява екран с няколко полета, отговарящи за обявата. След като търговецът попълни полетата и избере в коя категория иска да добави обявата, той цъка на бутона „Създай“, което добавя новонаправената обява към респективаната й категория.

Освен преглеждане и създаване на нови обяви, търговецът който е създал дадена обява може да я изтрие, от екрана с допълнителна информация за обявата, както и да я редактира от същия екран. При редакция търговецът бива препратен към екрана за създаване на нова обява, но този път полетата са попълнени с вече съществуващата информация, която той да промени. След като търговецът завърши редакцията на обявата трябва да натисне бутона „Запази“, който запазва новите полета и преработва вече съществуващата обява.

# Заключение

В дипломната работа бяха разработени сървър и мобилно приложение, които да работят взаимно. Мобилното приложение предоставя потребителски интерфейс за потребителите, а сървъра запазва цялата информация подадена и използвана от потребителите. Приложението е опростено разделяйки потребителите на два типа, което значително ги улеснява в целите им.

Допълнително може да се добави функционалността за оценка на даден търговец, която да влияе на това по какъв начин се изброяват обявите при избор на категория. Търговец с по-висока оценка означава коректност и добът продукт.

Друга функционалност която е планирана за бъдеща реализация е добавяне на потребители в списък „Чести клиенти“ от търговеца, като всеки потребител в този списък получава отстъпка на обявите създадени от търговеца.

# Източници

<https://futurestud.io/tutorials/retrofit-getting-started-and-android-client>

<https://developer.android.com/studio/intro/index.html>

https://github.com/bumptech/glide

<https://auth0.com/blog/json-web-token-signing-algorithms-overview/>

https://tools.ietf.org/html/rfc7519

https://en.wikipedia.org/wiki/JSON\_Web\_Token

https://www.androidhive.info/2016/04/android-glide-image-library-building-image-gallery-app/

http://www.baeldung.com/rest-with-spring-series/

https://www.tutorialspoint.com/struts\_2/basic\_mvc\_architecture.htm

http://www.tutorialsteacher.com/mvc/mvc-architecture

https://www.b4x.com/b4a.html

Светлин Наков и колектив – Въведение в програмирането с Java

СЪДЪРЖАНИЕ

[УВОД 4](#_Toc507528611)

[ГЛАВА I 5](#_Toc507528612)

[1.1 Приложения за търговия на стоки 5](#_Toc507528613)

[1.1.1 OLX 5](#_Toc507528614)

[1.1.2 Bazar.bg 7](#_Toc507528615)

[1.2 Основни среди за разработка на Android приложения 9](#_Toc507528616)

[1.2.1 Eclipse IDE 9](#_Toc507528617)

[1.2.2 Android Studio 10](#_Toc507528618)

[1.2.3 B4A 11](#_Toc507528619)

[ГЛАВА II 12](#_Toc507528620)

[2.1 Изисквания към дипломната работа 12](#_Toc507528621)

[2.1.1 Traders Mobile 12](#_Toc507528622)

[2.1.2 Traders Server 13](#_Toc507528623)

[2.2 Избор на език за програмиране 13](#_Toc507528624)

[2.3 Избор на среди за разработка 15](#_Toc507528625)

[2.3.1 Избор на среда за разработка на Android приложението 15](#_Toc507528626)

[2.3.2 Избор на среда за разработка на сървъра 15](#_Toc507528627)

[2.4 Избор на архитектура на приложението 16](#_Toc507528628)

[2.5 Избор на база данни 18](#_Toc507528629)

[2.5.1 Структура на база данни – артикул 19](#_Toc507528630)

[2.5.2 Структура на база данни – Категория 20](#_Toc507528631)

[2.5.3 Структура на база данни – Мерна Единица 20](#_Toc507528632)

[2.5.4 Структура на база данни – Потребител 21](#_Toc507528633)

[2.5.5 Структура на база данни – Склад 22](#_Toc507528634)

[2.5.6 Структура на база данни – Обект 23](#_Toc507528635)

[2.5.7 Структура на база данни – Свойства 23](#_Toc507528636)

[ГЛАВА III 24](#_Toc507528637)

[3.1 Основни модели 24](#_Toc507528638)

[3.1.1 Модел Потребител 24](#_Toc507528639)

[3.1.2 Модел Свойства 25](#_Toc507528640)

[3.1.3 Модел Обекти 25](#_Toc507528641)

[3.1.4 Модел Склад 26](#_Toc507528642)

[3.1.5 Модел Мерна единица 27](#_Toc507528643)

[3.1.6 Модел Категория 28](#_Toc507528644)

[3.1.7 Модел Артикул 29](#_Toc507528645)

[3.2 Създаване на обява 30](#_Toc507528646)

[3.2.1 Retrofit 30](#_Toc507528647)

[3.3 Зареждане на данните в мобилното приложение от базата данни 32](#_Toc507528648)

[3.3.1 Glide 33](#_Toc507528649)

[3.4 Защита JWT 34](#_Toc507528650)

[3.5 Изтриване на обява 36](#_Toc507528651)

[3.6 Промяна на обява 36](#_Toc507528652)

[ГЛАВА IV 37](#_Toc507528653)

[Ръководство на потребителя 37](#_Toc507528654)

[4.1 Traders 37](#_Toc507528655)

[4.2 Мобилно приложение Traders 37](#_Toc507528656)

[4.2.1 Инсталация 37](#_Toc507528657)

[4.2.3 Вход 39](#_Toc507528658)

[4.2.4 Категории 40](#_Toc507528659)

[4.2.5 Профил 41](#_Toc507528660)

[4.2.6 Обяви 42](#_Toc507528661)

[Заключение 44](#_Toc507528662)

[Източници 45](#_Toc507528663)