ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

Согласовано

Академический руководитель ОП
Системная и программная инженерия
профессор ДПИ

_____ Д.В. Александров

"____"_____ 2018 г

Утверждаю

Академический руководитель
образовательной программы
«Программная Инженерия»
профессор, канд. техн. наук

_____ В. В. Шилов

"____"_____ 2018 г

Клиент-Серверное Android-Приложение для Управления Скидками в Розничных Сетях

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.506900 81 01-1

Студент группы БПИ 151 НИУ-ВШЭ
_____ Куприянов К. И.
"____"_____ 2018 г

2018

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

УТВЕРЖДЕНО
RU.17701729.506900 81 01-1

Клиент-Серверное Android-Приложение для Управления Скидками в Розничных Сетях

Пояснительная записка

RU.17701729.506900 81 01-1

Листов 20

2018

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

1. Аннотация

В данном программном документе приведена пояснительная записка к программе “Клиент-Серверное Android-Приложение для Управления Скидками в Розничных Сетях”. В данном программном документе, в разделе «Введение» указано наименование программы, краткое наименование программы и документы, на основании которых ведется разработка. В разделе «Назначение и область применения» указано функциональное назначение программы и краткая характеристика области применения программы. В данном программном документе, в разделе «Технические характеристики» содержатся следующие подразделы:

- Постановка задачи на разработку программы;
- Описание алгоритма и функционирования программы с обоснованием выбора схемы алгоритма решения задачи и возможные взаимодействия программы с другими программами;
- Описание и обоснование выбора состава технических и программных средств

Так же к документу прикреплены приложения. Настоящий документ разработан в соответствии с требованиями:
1) ГОСТ 19.101-77 Виды программ и программных документов [4];
2) ГОСТ 19.102-77 Стадии разработки [5];
3) ГОСТ 19.103-77 Обозначения программ и программных документов [6];
4) ГОСТ 19.104-78 Основные надписи [7];
5) ГОСТ 19.105-78 Требования к программным документам [8];
6) ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению [10].

Изменения к данной Пояснительной Записке оформляются согласно ГОСТ 19.603-78 [11]. Перед прочтением данного документа рекомендуется ознакомиться с терминологией, приведенной в Приложении 1 настоящей Пояснительной Записки.

Содержание

2. Введение

2.1. Наименование программы

Наименование программы на русском: “Клиент-Серверное Android-Приложение для Управления Скидками в Розничных Сетях”.
Наименование на английском: “The Client-Server Android Application for Managing the Products’ Discounts in Retail Networks”.

2.2. Краткая характеристика

Программа предназначена для пользователей смартфонов на базе платформы Android. Цель работы - создать удобное приложения для составления списков покупок, добавления в них любых товаров (даже тех, которых нет в магазине), отслеживания акций, а так же просмотра всех акций в нескольких магазинах. Это позволит пользователям экономить свои средства на ежедневных покупках и быть в курсе актуальных акций магазинов. Серверная часть приложения должна представлять из себя web-crawler для сбора данных с сайтов розничных сетей. Должен быть реализован механизм взаимодействия с клиентами посредством обменивания файлами в формате JSON.
Главной чертой данного приложения является его лёгкая, быстрая масштабируемость и модульность программного кода.

3. Назначение разработки

3.1. Функциональное назначение

К функциональным возможностям программы относятся: просмотр списка актуальных акций, составление и редактирование списков покупок, регистрация и вход в аккаунт пользователей, возможность совместного использования списков покупок между пользователями. К функциональным возможностям серверной части относится сбор актуальных акций магазинов посредством crawling'а (см. терминологию) сайтов торговых сетей и отправки запросов на сервер приложения.

3.2. Эксплутационное назначение

Программа предназначена для запуска на мобильных устройствах операционной системы Андроид. Продукт является приложением “на стыке онлайна и оффлайна”, позволяющим планировать покупки в магазине с учетом актуальных скидок. Программа даёт пользователю возможность иметь несколько списков одновременно и добавлять в них любые товары; искать и подбирать интересующие его товары по сниженной цене, и, таким образом, экономить средства.

4. Технические характеристики

4.1. Постановка задачи на разработку программы

Программа должна соответствовать требованиям, представленным в Техническом Задании.

Задачи работы (Андроид-клиент):

1. Реализовать возможность просмотра списка доступных магазинов с акционными товарами
2. Реализовать представление текущих акций для конкретного магазина:
   (a) В виде общего списка
   (b) По категориям
3. Реализовать постепенную загрузку товаров магазинов (по страницам) для экономии трафика и меньшей нагрузкой на мобильное устройство
4. Реализовать регистрацию через мобильное приложение
5. Реализовать вход в аккаунт через мобильное приложение
6. Реализовать возможность смены аккаунта
7. Реализовать возможность создания списков покупок с разными названиями
8. Реализовать возможность удаления списка покупок
9. Реализовать возможность добавления товара в список покупок
10. Реализовать возможность удаления товара из списка покупок
11. Реализовать возможность добавления в список покупок пользовательских товаров, котороых нет в магазине (см. терминологию)
12. Реализовать возможность просмотра подобранных программой товаров согласно запросу пользователя
13. Реализовать добавление подобранных товаров в список покупок
14. Реализовать предварительное отображение элементов каждого списка покупок до их открытия
15. Реализовать отображение всплывающих подсказок при долгом нажатии на элементы управления button (кнопка)
16. Реализовать перенаправление в настройки сети для последующего включения интернета при отсутствии интернет-соединения
17. Реализовать отображение индикатора процесса загрузки данных с сервера
18. Реализовать обучающий фрагмент в разделе help (см. терминологию), содержащий руководство пользователя по управлению программой

Задачи работы (серверная часть):

1. Реализовать crawling веб-страниц для сбора актуальной информации об акционных товарах
2. Реализовать добавление товаров в базу данных посредством отправки запросов REST API (REST API и база данных реализованы напарником)

3. Реализовать запись акционных товаров со всех магахинах в JSON файл
4. Реализовать email уведомления администраторам сервиса об ошибках и неполадках в работе сервера

4.2. Описание алгоритма и функционирования программы

4.2.1. Описание построения Андроид приложения

В андроид приложении задумано 4 активности (см. терминологию) и 2 фрагмента (см. терменологию)

1. **LoginActivity** Рис 1. Активность, формирующая интерфейс входа пользователя в систему. Является самой первой ативностью, которая загружается всегда при запуске приложения, но если пользователь до этого момента уже входил в свой аккаунт, его данные сохранены в shared preferences (см. терминологию) и эта активность опускается. В этой активности 2 поля для ввода текста (логина и пароля) и 3 кнопки (войти, зарегистрироваться и продолжить использование без входа в аккаунт);
2. **RegisterActivity** Рис 2. Активность для регистрации пользователя. Имеет 2 поля для ввода текста (логина и пароля), и 2 кнопки: “Зарегистрироваться” (для новых пользователей) и “Войти” (для пользователей, уже имеющих аккаунт). Пользователь также может вернуться в активность для входа в систему посредством нажатия систесной кнопки “назад”;

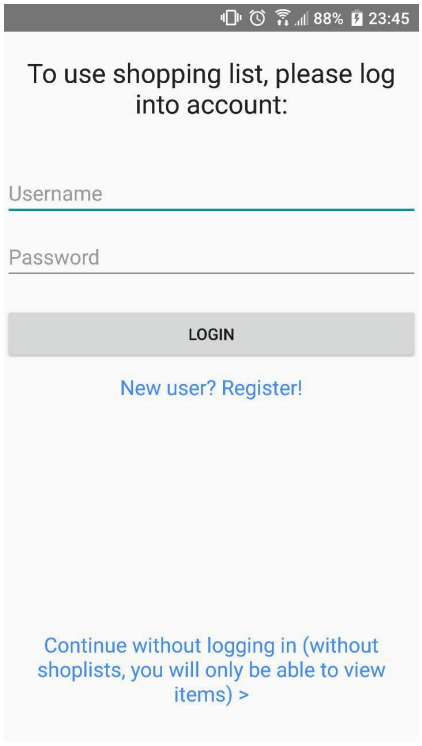

Рис. 1: Активность для входа в аккаунт пользователя

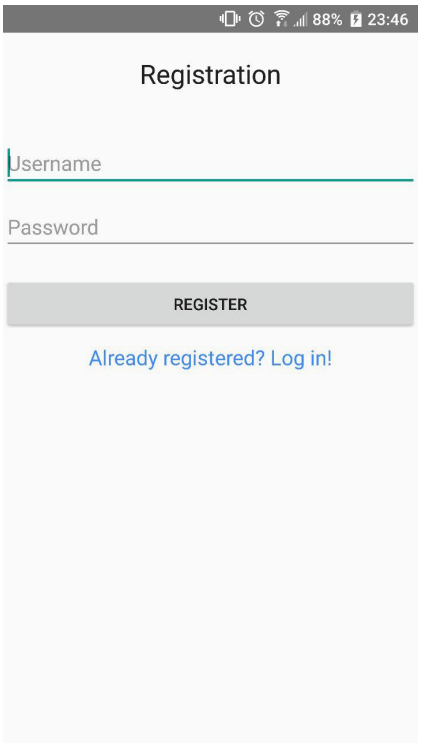

Рис. 2: Активность для регистрации аккаунта пользователя

3. **MainActivity** Главная активность, содержащая 2 дочерних фрагмента, toolbar и BottomNavigationView (см. терминологию).
   Фрагменты:
   (a) **HomeFragment** Рис 3. Фрагмент, содержащий список всех товаров. Список может быть отфильтрован по магазинам и по категориям. Содержит toolbar, в котором есть выпадающий список для выбора магазина, кнопка для выхода/входа из/в аккаунт(а), и меню опций;
   (b) **ShopListsPreviewFragment** Рис 4. Фрагмент, содержащий список всех списков покупок. Списки покупок упорядочены в виде карточек с предпросмотром в них добавленных товаров магазина и пользовательских товаров. Нажатие по одному из них открывает активность ShopListActivity, содержащую подробную информацию об элементах списка покупок. Долгое нажатие по одному из списков вызовет диалоговое окно, в котором спросится подтверждение намерения удалить данный список покупок. В элементе toolbar в этом фрагменте есть кнопка для входа/выхода в/из аккаунт(а), и кнопка для добавления нового списка покупок. Использование фрагмента не доступно неавторизованным пользователям;

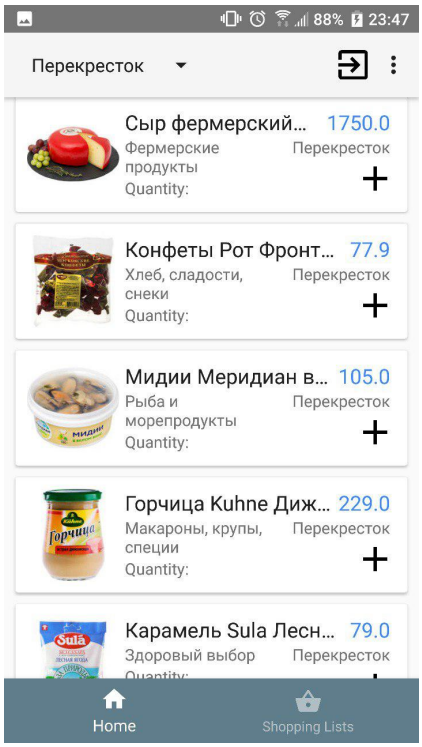

Рис. 3: MainActivity. Фрагмент со списком товаров магазинов

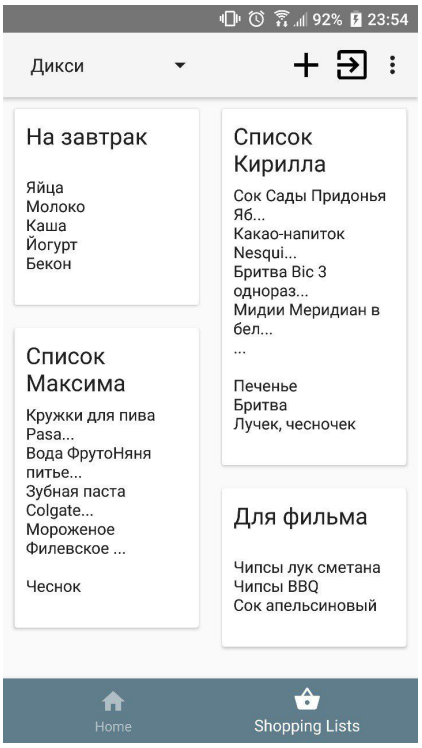

Рис. 4: MainActivity. Фрагмент со всеми списками покупок

4. **ShopListActivity** Рис 5, рис 6. Активность, содержащая подробную информацию о конкретном списке покупок. В элементе toolbar отображается кнопка “назад”, название списка покупок, и итоговая сумма по всем товарам, в стандартной для товаров валюте. В центре активности расположен список товаров магазина, а под ним - список пользовательских товаров. В свою очередь, каждый пользовательский товар содержит список подобранных товаров под данный из магазинов. Использование активности не доступно неавторизованным пользователям

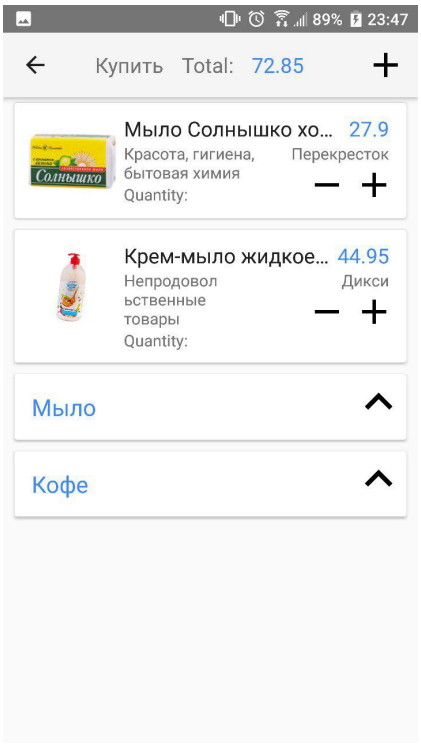

Рис. 5: ShopListActivity

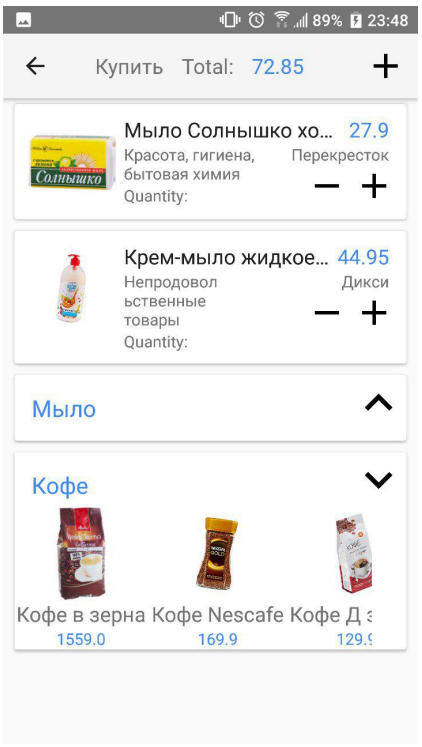

Рис. 6: ShopListActivity. Показаны соотретствующие пользовательскому товару товары их магазинов

4.2.2. Описание построения crawler'а

В crawler'е веб-страниц магазинов предусмотрены следующие модули: spiders/dixy_spider, spiders/perekrestok_spider, text_processor, dixy_selectors, perekrestok_selectors, settings, config, notify и pipelines.

1. **dixy_spider** описывает класс DixySpider, унаследованный от Scrapy.Spider. Отвечает за получение исходного кода страницы магазина "Дикси", его парсинга при помощи "модульных" css селекторов (см. терминологию), перехода по страницам на сайте. Формирует объекты классы DixyItem и передаёт далее модулю pipelines на обработку;
2. **perekrestok_spider** описывает класс PerekrestokSpider, функционал идентичен DixySpider, но действует на сайте магазина "Перекресток";
3. **text_processor** содержит функции обработки текста при помощи регулярных выражений и базовых операций со строками;
4. **dixy/perekrestok selectors** содержат соответствующие xpath селекторы для веб-страниц магазинов. Выделение этой логики в отдельный модуль - корневая причина модульности приложения. В случае изменения дизайна сайта, кроулинг которого происходит, администратору придётся изменить лишь одну строчку для одного элемента на странице;
5. **settings, config** содержат настройки и конфигурацию кроулера;
6. **pipelines** содержит методы для обработки объектов DixyItem и PerekrestokItem. Мо-

дуль так же совершает POST запрос на сервер приложения для добавления товара в базу данных, и сохраняет все объекты в JSON файл

4.3. Ежедневное обновление акций

Процесс ежедневного обновления акций необходимо проводить, чтобы акции в приложении всегда были актуальными.

На сервере запущена cron job (см. терминологию), котрая каждый день в 2:00 ночи в 6:00 утра запускает процесс кроулинга акций во всех магазинах:

```
00 2,6 * * * cd /home/hes/Projects/crawler/easysales && ./crawl_all.sh > /home/hes/crawler.log 2>&1
```

Скрипт crawl_all.sh отвечает за поочередный запуск всех spider'ов. Далее, по цепочке, спайдеры собирают данные с сайтов при помощи xpath селекторов, используют модуль text_processor для очистки собранных данных от non-breaking space символов, длинных пробелов и других символов. Формируются <Shop>Items, которые затем отправляются в модуль pipelines. Классы <Shop>Items, процесс добавления Item'а в базу и модель с селекторами для магазина "Перекресток" описаны ниже.

Классы DixyItem и PerekrestokItem унаследованы от EasySalesItem, базового товара приложения. Его поля всегда заполнены, то есть не могут иметь значения null.

```
class EasySalesItem(scrapy.Item):
    name = scrapy.Field()
    newPrice = scrapy.Field()
    imageUrl = scrapy.Field()
    shopId = scrapy.Field()
    crawlDate = scrapy.Field()


class DixyItem(EasySalesItem):
    category = scrapy.Field()
    oldPrice = scrapy.Field()
    discount = scrapy.Field()
    dateIn = scrapy.Field()
    dateOut = scrapy.Field()
    condition = scrapy.Field()


class PerekrestokItem(EasySalesItem):
    category = scrapy.Field()
    oldPrice = scrapy.Field()
```

Отправка объекта класса Item в базу данных происходит при помощи API запроса на сервер:

```
class DataBaseWriterPipeLine(object):

    def process_item(self, item, spider):
        requests.post(const['ADD_ITEM_API'],
```

```
            data=json.dumps(dict(item)),
            headers=const['REQUEST_HEADERS'])
    return item
```

Xpath селекторы для магазина “Перекресток”. Модульность и простота использования написанного фрагмента кода достигается тем, что при изменении дизайна сайта магазина, возникает необходимость исправить всего несколько строчек в приведённом фрагменте. Таким образом, частично решается проблема всех кроулеров - поломка, в случае изменения дизайна сайта. А поскольку большинство кроулеров не используют такой модульный подход, времени и сил на починку уходит гораздо больше, так как приходится править множество файлов во многих местах.

```
# Start urls.
#
URLS = ['https://www.perekrestok.ru/catalog']
URL_CORE = 'https://perekrestok.ru'

# Categiries.
#
CATEGORIES = '//a[@class="xf-catalog-categories__link"]/@href'
CATEGORIES_TEXT = '//span[@class="xf-catalog-categories__text"]/text()'
POST_CATEGORY = '//span[@class="xf-breadcrumbs__current"]/text()'
CURRENT_CATEGORY = '//h1[@class="xf-caption__title"]/text()'

ROOT_NODE = '//ul[@id="catalogItems"]'
ITEM = 'li/div[contains(@class, "xf-product")]'

# Item attributes.
#
NAME = 'div[@class="xf-product__title xf-product-title"]/a[@class="xf-product-\
title__link js-product__title"]/text()'
IMG = 'figure/a/img/@data-src'
NEW_PRICE = 'div/div[contains(@class, "xf-product-cost__current")]/@data-cost'
OLD_PRICE = 'div/div[contains(@class, "xf-price xf-product-cost__prev")]/@data-cost'

# Pagination selector.
#
# next_page = '//li[@class="next"]/a/@href'
NEXT_PAGE = '//a[contains(@class, "xf-paginator__item js-paginator__next")]/@href'
```

5. Технико-экономические показатели

5.1. Предполагаемая потребность

“Клиент-Серверное Android-Приложение для Управления Скидками в Розничных Сетях” может быть использована в потребительской сфере. Программу могут использовать как люди, нуждающиеся в экономии средств посредством покупки более дешёвых товаров, так и люди, желающие эффективно и быстро работать со списками покупок.

5.2. Экономические преимущества разработки

На момент принятия решения о написании данного продукта во Всемирной сети Интерент был лишь 1 аналог: приложение “Едадил”. Преимущества данной разработки перед конкурентом:

1. Лёгкий, не перегруженный излишним функционалом дизайн
2. Лучший сбор картинок (при помощи кроулинга)
3. Возможность создания неограниченного количества списков покупок с разными названиями
4. Возможность добавления в списки покупок товаров, которых нет в наличии в магазинах

6. Источники, используемые при разработке

Список литературы

[1] *Android Developers references and guides*. 2018. URL: https://developer.android.com/index.html.

[2] *Google's material design color guidline*. 2015. URL: https://material.io/guidelines/style/color.html.

[3] *Scrapy 1.5 documentation*. 2018. URL: http://scrapy.readthedocs.io/en/latest/index.html.

[4] Единая Система Программной Документации. *ГОСТ 19.101-77 Виды программ и программных документов*. ИПК Издательство стандартов, 2001.

[5] Единая Система Программной Документации. *ГОСТ 19.102-77 Стадии разработки*. ИПК Издательство стандартов, 2001.

[6] Единая Система Программной Документации. *ГОСТ 19.103-77 Обозначения программ и программных документов*. ИПК Издательство стандартов, 2001.

[7] Единая Система Программной Документации. *ГОСТ 19.104-78 Основные надписи*. ИПК Издательство стандартов, 2001.

[8] Единая Система Программной Документации. *ГОСТ 19.106-78 Требования к программным документам*. ИПК Издательство стандартов, 2001.

[9] Единая Система Программной Документации. *ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению*. ИПК Издательство стандартов, 2001.

[10] Единая Система Программной Документации. *ГОСТ 19.404-79 Пояснительная записка. Требования к содержанию и оформлению*. ИПК Издательство стандартов, 2001.

[11] Единая Система Программной Документации. *ГОСТ 19.603-78 Общие правила внесения изменений*. ИПК Издательство стандартов, 2001.

7. Приложение 1. Терминология

7.1. Терминология

Активность (Activity) – Activity — это компонент приложения, который выдает экран, и с которым пользователи могут взаимодействовать для выполнения каких-либо действий, например набрать номер телефона, сделать фото, отправить письмо или просмотреть карту.

Фрагмент (Fragment) – Фрагмент (класс Fragment) представляет поведение или часть пользовательского интерфейса в операции (класс Activity). В одной активности может быть несколько фрагментов.

Shared Preferences – Файл в системе, содержащий информацию в виде пар “ключ - значение”.

Toolbar – Верхняя часть приложения, содержащая наиболее часто используемые функциональные элементы, например кнопка “назад” или “создать”.

Bottom Navigation View – Элемент навигации, представляющий из себя несколько кнопок, ведущих на разные фрагменты и расположенный снизу экрана устройства.

Xpath selectors – Язык для выделения из *ML (XML, HTML) кода его элементов, тэгов и аттрибутов.

Cron job – Сервис в системах GNU/Linux и UNIX, позволяющий ставить на повторяющиеся циклы исполнение пользовательских команд.

crawler – Программный модуль, работающий в фоне и производящий сбор данных с сайтов указанных магазинов, с последующей отправкой их на сервер в формате JSON

Пользовательский товар – Товар, представленный в виде текста, имеющий в себе массив товаров, подходящих при сопоставлении названий к данному. Пример. Пользоваельский товар “Сок” имеет массив сопоставившихся товаров [Сок Добрый 1л Яблоко; Сок J-7 апельсин с мякотью].

Пользовательская сессия, user session – Начинается с момента входа пользователя в систему (log in), и завершается при выходе (log out).

Spider – Часть crawler’a, отвечающая за непосредственный сбор информации с веб-страниц, переход между страницами и дальнейшую отправку собранных данных другим модулям crawler’a.

Non-breaking space – Специальный пробельный символ, предотвращающий автоматический разрыв строки в месте, где он стоит.

8. Приложение 2. Описание классов

Ниже приведены описания классов для Андроид клиента (Java)

Классы и интерфейсы Java:

8.1. LoginActivity

Поля:

```
private EditText etUsername;
private EditText etPassword;
private TextView tvRegister, tvHack;
private Button btnLogin;
private Button.OnClickListener btnLoginListener;
private SharedPreferences sharedPrefs;
private Map<String, String> userData;
```

Класс, отвечающий за отображение LoginActivity, проверку введённых данных и отправку запроса авторизации на сервер.

8.2. MainActivity

Поля:

```
private boolean doubleBackToExitPressedOnce = false;
private SwipeRefreshLayout swipeRefreshLayout;
private static final String TAG_FRAGMENT_ONE = "fragment_one";
private static final String TAG_FRAGMENT_TWO = "fragment_two";
private Shop selectedShop = null;
private View btnAdd;
private View btnLoginLogout;
public boolean homeActive = true;
public int currentPage = 1;
public String selectedCategory = "";
public int totalItemsCount;
public Parcelable itemsFragmentState;
public Parcelable shopListsPreviewFragmentState;
public ItemAdapter adapter;
public ShopListsPreviewAdapter shopListsPreviewAdapter;
public FetchData fetchData;
public List<Shop> shops;
public RequestQueue queue;
private FragmentManager fragmentManager;
public List<String> categories;
private int currentFragmentId;
```

Основной класс программы, хранит состояния фрагментов, чтобы состояния прокрутки оставалась неизменной при переходе по фрагментам; запускает процессы отображения

всех основных графических элементов, отвечает за логику перемещения внутри приложения и определяет поведение кнопки “назад”.

8.3. RegisterActivity

Поля:

```
private EditText etUsername, etPassword;
private Button btnRegister;
private Map<String, String> userData;
```

Класс, отвечающий за отображение RegisterActivity, проверку введённых данных и отправку запроса авторизации на сервер.

8.4. ShopListActivity

Поля:

```
RecyclerView.LayoutManager layoutManager;
RecyclerView rvShopList;
SwipeRefreshLayout swipeRefreshLayout;
public ItemAdapter adapter;
public ShopList selectedShopList;
public FetchData fetchData;
public TextView tvTotalPrice;
private View btnAdd;
private TextView tvShopListName;
```

Класс, отвечающий за отображение ShopListActivity, загрузку списков покупок с сервера.

8.5. ItemAdapter

Поля:

```
private List<Item> items;
private List<Item> tmpItems;
private ArrayList<Item> itemsCopy;
private Context context;
private Dialog itemFullPreview;
private SparseBooleanArray expandState;
private boolean type1Downloaded = false;
private boolean type2Downloaded = false;
```

Класс, отвечающий за отображение каждого индивидуального Item’а.

8.6. ShopListPreviewAdapter

Поля:

```
public List<ShopList> shopLists;
private Context context;
```

Класс, отвечающий за отображение каждого индивидуального ShopList'a.

8.7. HomeFragment

Поля:

```
private RecyclerView rvItemList;
private RecyclerView.LayoutManager layoutManager;
```

Класс, отвечающий за отрисовку главного фрагмента и toolbar'a.

8.8. ShopListsPreviewFragment

Поля:

```
RecyclerView.LayoutManager layoutManager;
RecyclerView rvShopLists;
```

Отображает "превью" всех пользовательских предметов и товаров из магазина в виде карточек

8.9. InternetUtil

Класс, содержащий статический метод для проверки работы интернет-сети.

8.10. JSONUtil

Класс, формирующий хэдер JSON запроса.

8.11. APIRequests

Имеет статическую фабрику для получения Request Handler'a. Далее все действия проводит RH. он добавляет к запросу URL, ReqponceListener и ErrorListener

8.12. Config

Поля:

```
public static final String URL_CORE = "http://gcsales.ru/";
public static final String URL_LOGIN = URL_CORE + "auth/login/";
public static final String URL_REGISTER = URL_CORE + "auth/register/";
public static final String URL_SALES_SHOP = URL_CORE + "api/shops/";
public static final String URL_SHOPLIST = URL_CORE + "api/shoplist/";
public static final String URL_SHOPLISTS_PREVIEW = URL_CORE + "api/shoplist?mode=preview";
public static final String URL_SHOPLISTS = URL_CORE + "api/shoplist?mode=full";
public static final String URL_ITEMS_ON_PAGE = "&page=";
public static final String URL_ITEMS_IN_CATEGRY = "?category=";
public static final String URL_CATEGORIES = "categories";
public static final String URL_SL_ADD_ITEM = "additem?id=";
public static final String URL_SL_ADD_CUSTOM_ITEM = "additem?custom=";
public static final String URL_SL_DELETE_ITEM = "deleteitem?id=";
public static final String KEY_USERNAME = "username";
public static final String KEY_PASSWORD = "password";
public static final String TAG_VOLLEY_ERROR = "VOLLEY";
public static final String REQUESTS_CONTENT_TYPE = "application/json; charset=utf-8";
public static final String BAD_API_AUTH_RESPONSE = "Wrong username/password.";
public static final String SH_PREFS_NAME = "easy_sales_token";
public static final String KEY_TOKEN = "user_token";
public static final String DEF_NO_TOKEN = "NULL";
public static final String KEY_ITEMS = "all_items";
public static final String KEY_SHOPLIST_ITEMS = "shoplist_items";
public static final String KEY_CURRENT_SHOPLIST = "com.hes.easysales.easysales.ShopList.";
public static final int MAX_LEN_PREVIEW = 20;
public static final int MAX_ITEMS_PREVIEW = 6;
```

Класс с общими для всех даными.

8.13. EndlessRCVScrollListener

Поля:

```
private LinearLayoutManager linearLayoutManager;
private WeakReference<Activity> actvityRef;
```

Класс, “слушающий” когда элемент RecyclerView дойдёт долистается конца. И подгружает ещё больше товаров.

8.14. FetchData

Поля:

```
private ProgressBar pbLoading;

// To prevent the leak of Context.
//
```

```
private WeakReference<Activity> activityRef;
private WeakReference<SwipeRefreshLayout> swipeRefreshLayoutRef;
```

Класс, который по мере возможности загружает всю необходимтую информацию.о

8.15. Item

Поля:

```
private long id;
private String name;
private String category;
private String imageUrl;
private double oldPrice;
private double newPrice;
private String discount;
private String dateIn;
private String dateOut;
private String condition;
private Shop shop;
// These fields are initialized explicitly using setters!
// Factory method keeps them default.
// However, Parcel keeps them for future reuse.
//
private boolean expandable = false;
private boolean matched = false;
private List<Item> matchingItems = new ArrayList<>();
```

Класс, реализующий POJO item Item.

8.16. Shop

Поля:

```
private int id;
private String alias;
private String name;

private List<String> categories;
```

Класс, реализующий POJO item Shop.

8.17. ShopList

Поля:

```
private long id;
private String name;
private List<Item> items;
private List<Item> customItems;
```

Класс, реализуюций POJO item ShopList.

8.18. ItemClickListener

Метод:

```
public void onClick(View v, int postition, boolean isLongClick);
```

Интерфейс, определяющий поведение Item'ов при нажатии на них.

Лист регистрации изменений

Изм.	Номера листов (страниц)				Всего листов (страниц) в докум.	№ докум.	Входящий № сопроводительного докум. и дата	Подпись	Дата
	измененных	замененных	новых	аннулированных					