ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботи

на ступінь вищої освіти бакалавр

на тему: **«РОЗРОБКА АНДРОЇД ДОДАТКУ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ MYSKATEBOARDTRICKSEDUCATION»**

Виконав: студент 4 курсу, групи ПД-44

спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Оверченко Є.Ю.

(прізвище та ініціали)

Керівник Яскевич В.О.

(прізвище та ініціали)

Рецензент

(прізвище та ініціали)

Нормоконтроль

(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти - «Бакалавр»

Спеціальність -121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

Інженерії програмного забезпечення

_____О.В. Негоденко

«____»_____2021 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

ОВЕРЧЕНКО ЄВГЕНІЙ ЮРІЙОВИЧ

1.Тема роботи: «Розробка андроїд додатку для мобільних пристроїв Myskateboardtrickseducation»

Керівник роботи Яскевич Владислав Олександрович, кандидат технічних наук, доцент

затверджені наказом вищого навчального закладу від — «12» березня 2021 року №65.

2. Строк подання студентом роботи 01.06.2021

3. Вхідні дані до роботи:

3.1. Середовище розробки PyCharm Community Edition 2020.3.3

3.2. Алгоритм дії представленого мобільного додатку

3.3. Науково-технічна література, пов’язана з розробкою Андроїд

додатків

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1. Аналіз та огляд існуючих програмних рішень для трекінгу

4.2. Аналіз та виділення стадій розробки

4.3. Дослідження програмних засобів для розробки додатку

4.4. Розробити функціонал серверу, парсеру та клієнта

5. Перелік графічного матеріалу

5.1.1. Алгоритм дії представленого мобільного додатку

5.1.2. Програмні засоби реалізації

5.1.3. Огляд можливостей розробленого додатку

6. Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.21 – 22.04.21	
2	Аналіз існуючих прототипів	22.04.21 – 23.04.21	
3	Дослідження програмних засобів	23.04.21 – 01.05.21	
4	Моделювання об’єкту проектування	01.05.21 – 05.05.21	
5	Розробка функціоналу бота	05.05.21 – 07.05.21	
6	Вступ, висновки, реферат	07.05.21 – 08.05.21	
7	Розробка презентації застосунку	08.05.21 – 24.05.21	
8	Попередній захист роботи	25.05.21	

Студент

Керівник роботи

РЕФЕРАТ

Текстова частина бакалаврської роботи 79с., 24 рис., 25 таблиць, 20 джерел.

Ключеві слова: мобільний додаток, Андроїд, парсинг, трекінг, скейт.

Об'єкт дослідження – поліпшення функціоналу навчання та трекінгу трюків на скейтборді в процесі навчання.

Предмет дослідження – Андроїд додаток для трекінгу прогресу навчання трюків на скейтборді «MySkateboardTricksEducation».

Мета роботи – розробка мобільного додатку на базі Андроїд.

Методи дослідження – методи теорії інформації, методи структурного аналізу і проектування, методи розробки програмного забезпечення, методи тестування, валідації та верифікації програмного забезпечення.

Слід зазначити що даний дипломний проект є актуальним та може розвиватися в майбутньому.

Дана система направлена на полегшення навчання, складання списку трюків та загалом для полегшення життя людям, а саме для тих хто хоче оволодіти скейтбордом.

Галузь використання – додатком може користуватися будь-яка людина, яка бажає навчитися трюкам на скейтборді, або поліпшити свої навички та відслідковувати їх.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ – Програмне Забезпечення

ОС - Операційна система

RFC - від англ. Request for Comments Запит коментарів, Прохання прокоментувати

TCP - від англ Transmission Control Protocol Протоко́л керува́ння передачею

IP - від англ. Internet Protocol address ідентифікатор унікальний числовий номер мережевого рівня, який використовується для адресації комп'ютерів чи пристроїв у мережах

RPC - від англ. Remote Procedure Call віддалений виклик процедур

D-COM - від англ. Distributed Component Object Model модель об'єктів

DHCP - від англ. Dynamic Host Configuration Protocol — протокол динамічної конфігурації вузла

REST – від англ. Representational state transfer, стиль архітектури програмного забезпечення для розподілених систем

SMTP - від англ. Simple Mail Transfer Protocol, Простий Протокол Пересилання Пошти

NTP - від англ. Network Time Protocol, Мережевий протокол час

HTTP - від англ. Hyper Text Transfer Protocol, протокол передачі даних

LDAP - англ. Lightweight Directory Access Protocol - полегшений протокол доступу до списків

SQL - від англ. Structured Query Language - мова структурованих запитів

GUI – Graphical User Interface

API – Application Programming Interface

UML – Unified Modelling Language

ВСТУП

Швидкий розвиток інформаційних технологій та вдосконалення комп'ютерних технологій призвели до їх глобальної інтеграції у всіх сферах людської діяльності. Сфера мереж супермаркетів не є винятком.

Сучасну молодь, поряд з підлітками і більш дорослими прихильниками активного способу життя все частіше можна побачити на скейті - навіть незважаючи на те, що скейтбординг все ще сприймається, як забава для дітей

Втім, уроки скейтбордингу є досить популярними в наші дні, прирівнюючи скейт до інших, більш поширених екстремальних видів спорту — як серед дорослих, так і серед дітей.

Навчання скейтбордингу. Сьогодні скейтбординг - не тільки спорт, спосіб пересування по місту, а й стиль життя. Якщо ви любите скейт так само як любимо його ми, чекаємо вас в нашій школі. Тренери Victory School навчать вас кататися на скейтборді, навчать вас не тільки правильному катанню і виконання різних трюків, але і філософії скейтбордингу. Чи безпечно, правильно, легко і в найкоротші терміни ви будете вже впевнено стояти на дошці.

Об’єктом дослідження є «Мобільний додаток для трекінгу особистого прогресу в скейтбордингу».

Додаток має дві основні вкладки - профіль та навчання.

У Вкладці «профіль» відображається особистий прогрес, тобто особистий рівень (початківець, любитель, продвинутий, про), кількість трюків які вивчені, та має бути можливість переглянути які трюки вже вивчені. Окрім цього відображається фото користувача,його нікнейм та стаж катання.

У Вкладці «навчання» мають бути 5 розділів (Основи, шовіти, фліпи, слайди та гранди, греби). Всі трюки розподіляються по певним розділам і вивчаються від легшого до складнішого, але не обов’язково, можно переходити вручну. Кожна сторінка з вивчення трюку має текстове поле, гіф з відображенням трюку та відео урок підтягнутий з ютубу.

1 ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ТРЕКІНГУ ОСОБИСТОГО ПРОГРЕСУ В СКЕЙТБОРДИНГУ

1.1 Огляд ПЗ конкурентів на ринку

Операційна система Android є інноваційною, яка з кожним роком набирає все більшу і більшу популярність. Популярна операційна система для телефонів і інших гаджетів створена на базі Linux, яка була і залишилася конкурентом Windows від компанії Microsoft. ОС Android - це відкрита продукція, для якого є в наявності велика кількість різноманітних програм в безмежному безкоштовному доступі. До того ж ОС безперервно розвивається і поліпшується.

Володарі і творці ОС Android це компанія Open Headset Alliance, де налічується близько 80 різних фірм, навіть Google. 1-е дітище з даною системою на борту було виставлено на загальний огляд напочатку 2008 року. Після чого послідували пропозиції від виробників мобільної техніки. Зараз Андроїд можливо зустріти не стільки на телефонах, але і на планшетних пристроях, а також і в фоторамках.

Основна частина програм для даної системи написана на мові Java. Завдяки опису операційної системи Android можна з'ясувати, як вона працює, які застосовуються двигуни і бібліотеки.

Починаючи з версії операційної системи під назвою Android 1.6 помітно перетворилася працездатність завантаження і покупки програм Play Market головну сторінку якого продемонстровано на рисунку 1.1. Як і раніше, всі додатки поділяються на 2 масштабні групи Програми та Розваги, а далі на підкатегорії. У переліку підкатегорій показуються самий часто завантаження контент будь-якого з розділів. Найактуальніше нововведення - в опис програми або ж вставлені скріншоти, які дають Вам можливість розцінити інтерфейс нового контенту. Це особливо необхідно при завантаженні

комерційних програм, хоча крім того знадобиться і для приблизної оцінки дизайну нової заставки або гри, щоб не витрачати трафік і час даремно.

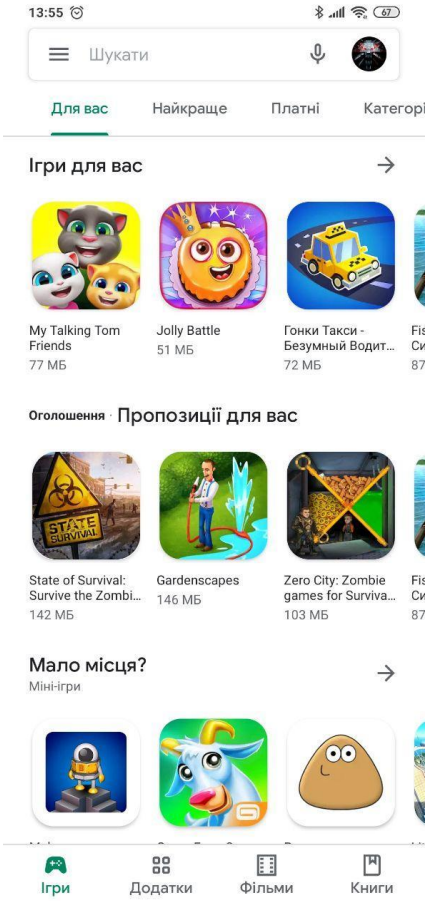


Рисунок 1.1 – головна сторінка Play market

Загалом, для вітчизняних користувачів, як і раніше доступні лише безкоштовні завантаження. Крім того спрацьовує система оцінки. Будь-який користувач, що завантажив програму.

1.2 Формування вимог до програмного трюку

Серверний додаток (сервер) запускається на комп'ютері, так само званому "сервер", при цьому при розгляді топології мережі, такий вузол називають "сервером". Даний сервер отримуватиме дані з парсера та передаватиме клієнту.

Основні функціональні вимоги сервера:

– Отримання запитів від клієнта;

– Обробка запитів від клієнта;

– Відправка інформації клієнту;

– Отримання інформації від парсера;

– Зберігання даних;

– Обробка даних від парсера;

– Отримання декілька запитів (багатопоточність);

– Формування каталогу трюків для клієнта.

Парсер ж в свою чергу моніторить сайти з знижка та оновлює дані раз в дві доби формує даними в JSON та відправляє серверу.

Основні функціональні можливості паресера:

– Пошук акційних пропозицій супермаркетів;

– Формування карток трюків;

– Оновлення актуальної інформації про товари;

– Стежити за змінами та повідомлювати сервер у виді блоку інформації;

– Формування бази даних трюків.

В свою чергу клієнт (Android) додаток для відстеження знижок і акцій в продуктових магазинах. Отримуватиме дані з серверу та відправлятиме запити на нього. Основні функціональні можливості додатку:

– каталогу акцій в зручному та доступному вигляді, робота зі списком трюків.

– продукт є додатком "на стику онлайну і оффлайна", що дозволяє планувати покупки в магазині з урахуванням актуальних знижок. До функціональних можливостей відноситься:

– програми відносяться: перегляд списку актуальних акцій, складання і редагування списку трюків, реєстрація користувачів, можливість спільного

використання акаунтів між користувачами, а так само підбір трюків відповідно до призначених для користувача запитами.

– програма призначена для запуску на мобільних пристроях операційної системи Android (Android-Додаток) .

– програма є додатком, що дозволяє користувачу шукати і підбирати потрібні йому товари за зниженою ціною, і, таким чином, економити кошти.

Існуючим аналогом даного трюку є додатки наведені вище рис.1 та рис В силу того, що це додаток також поширюється безкоштовно, даний програмний продукт буде економічно ефективним конкурентом.

Однак, деякі додатки набагато складніші у використанні, переповнений зайвим функціоналом і має важкий інтерфейс.

З вищесказаного випливає, що потрібно розробити зручний користувацький інтерфейс для осіб різнового вік, а саме можна виділити наступні фактори:

– Читабельність тексту;

– Зрозумілість у використанні;

– Легкий функціонал;

– Радісна гамма кольорів;

– Можливість зберігання списків у зручному форматі.

Відштовхуючись від поставлених завдань потрібно розбити розробку на три стадії, а саме:

1. розробка серверу та його функціоналу;
2. розробка парсера та його функціоналу;
3. розробка Android-додатку (клієнта).

Що спростить розробку та звиде до мінімізування помилок під час проектування.

1.3 Завдання і цілі сучасних ПЗ

З кожним роком сучасні технології розвиваються, розробка мобільних додатків також не стоїть на місці. Їх реалізація сприяє покращенню комунікативності, якості спілкування, вирішення різноманітних складних завдань. Хорошим професіоналам це приносить досить високий прибуток.

Розробка мобільних додатків є досить непростим та довгим процесом. Додатки розробляються для мобільних телефонів, планшетів, інших портативних пристроїв. Існує декілька основних етапів розробки, без яких процес буде неможливим.

По-перше, потрібно вибрати платформу (iPhone, Android, планшети, смартфони). Можна вибирати відразу декілька платформ, але для цього необхідно збільшувати бюджет.

По-друге, треба визначитися з основною метою створення додатку та його актуальністю.

Наступним етапом є створення макету додатку. Кожна деталь є досить важливою, адже упустивши одну, на перший погляд незначну, доведеться переробляти все.

Далі розробляється дизайн мобільного додатку. Він повинен враховувати, насамперед, головну ціль створення, цільову аудиторію. Якщо для бізнес-цілей,

то має бути більш стриманим, з помірними кольорами, якщо ж ігровий додаток – навпаки – яскравим, щоб привертати увагу. Спочатку розробляється дизайн перших трьох головних сторінок, які є основою для наступних. При цьому потрібно опрацьовувати кожну деталь.

Наступний етап – програмна розробка. Щоб додаток функціонував повноцінно, поєднуються між собою усі елементи: кнопки, екрани, іконки.

Тестування є не менш важливим етапом розробки. Проводиться з метою усунення помилок, які формуються в певну таблицю. Кожен додаток є унікальним, тому передбачити і уникнути багів досить важко. Необхідно

зробити усе для того, аби на кінцевому етапі споживач отримав ідеальний продукт.

Завершальним етапом є розміщення мобільного додатку. Існують спеціальні магазини для розміщення. Найбільш популярним є GooglePlay, Ovi Store та AppStore. Вибір потрібно здійснювати, враховуючи цільову аудиторію та техніку, якою вона користується. Існують різні типи мобільних додатків:

- контентні – створюються для поширення певної інформації;
- корпоративні – для вирішення різноманітних бізнес-цілей;
- ігрові – створюються з розважальною метою;
- сервісні – надання сервісних послуг (будильник та багато інших).

Також не потрібно забувати про те що потрібна підтримка додатку. Підтримка у вигляді серверу, який оброблятиме все на своїй стороні.

Тобто централізоване управління доступом до інформації. За допомогою таких широко відомих технологій, як обміну інформації по мережі, допомагає структурувати систему зберігання даних за рівнями доступу, в результаті розробляється система, де: кожен користувач має доступ тільки до певної інформації; на кожному ресурсі можливо збереженні тільки тих даних, які визначені політикою безпеки; є можливість протоколювання будь-яких подій доступу до інформаційних ресурсів; вся збережена інформація чітко впорядкована. Зручно всі дані зберігаються на сервері. Доступ до них для різних користувачів і груп користувачів можна обмежити, що зменшить ризик втрати критичних даних.

Для полегшення заповнення сервера інформацією є розробка парсера. Адже збір і аналіз інформації в Інтернеті займає багато часу, сил і ресурсів. Автоматизована програма парсер справляється з таким завданням швидше і легше. Вона протягом доби здатна «пролистати» більшу частину веб-контенту в Мережі в пошуку потрібних даних і проаналізувати їх.

Відповідно, за допомогою програми- парсера можна знаходити контент для наповнення і парсити веб сторінки сайтів супермаркетів для пошуку та витягування інформації з них та відправки клієнтам які в свою чергу будуть

зберігатися та оброблятися а сервері, щоб користувач зміг швидко їх витягувати.

1.4 Сучасна архітектура ПЗ

Поняття сервер і клієнт і закріплені за ними ролі утворюють програмну концепцію «клієнт-сервер». Модель клієнт-сервер - це ще один підхід до структурування ОС. У широкому сенсі модель клієнт-сервер передбачає наявність програмного компонента - споживача будь-якого сервісу - клієнта, і програмного компонента - постачальника цього сервісу - сервера. Взаємодія між клієнтом і сервером стандартизуется, так що сервер може обслуговувати клієнтів, реалізованих різними способами і, може бути, різними виробниками. При цьому головною вимогою є те, щоб вони запитували послуги сервера зрозумілим йому способом. Ініціатором обміну зазвичай є клієнт, який надсилає запит на обслуговування сервера, що знаходиться в стані очікування запиту. Один і той же програмний компонент може бути клієнтом по відношенню до одного виду послуг, і сервером для іншого виду послуг. Модель клієнт-сервер є скоріше зручним концептуальним засобом чіткого уявлення функцій того чи іншого програмного елемента в тій чи іншій ситуації, ніж технологією. Ця модель успішно застосовується не тільки при побудові ОС, але і на всіх рівнях програмного забезпечення, і має в деяких випадках більш вузький, специфічний сенс, зберігаючи, природно, при цьому всі свої загальні риси. Стосовно до структурування ОС ідея полягає в розбитті її на кілька процесів - серверів, кожен з яких виконує окремий набір сервісних функцій - наприклад, управління пам'яттю, створення або планування процесів. Кожен сервер виконується в режимі користувача. клієнт, яким може бути або інший компонент ОС, або прикладна програма, запитує сервіс, посилаючи повідомлення на сервер. Ядро ОС (зване тут мікроядром), працюючи в привілейованому режимі, доставляє повідомлення потрібного сервера, сервер виконує операцію, після чого ядро повертає результати клієнту

за допомогою іншого повідомлення. Для взаємодії з клієнтом (або клієнтами, якщо підтримується одночасна робота з декількома клієнтами) сервер виділяє необхідні ресурси між процесами взаємодії (колективна пам'ять, пайп, сокет, і т. П.) І очікує запити на відкриття з'єднання (або, власне, запити на наданий сервіс). Залежно від типу такого ресурсу, сервер може обслуговувати процеси в межах однієї комп'ютерної системи або процеси на інших машинах через канали передачі даних (наприклад,

Формат запитів клієнта і відповідей сервера визначається протоколом. Специфікації відкритих протоколів описуються відкритими стандартами, наприклад, протоколи Інтернету визначаються в документах RFC.

Залежно від виконуваних завдань одні сервери, при відсутності запитів на обслуговування, можуть простоювати в очікуванні. Інші можуть виконувати якусь роботу (наприклад, роботу зі збору інформації), у таких серверів робота з клієнтами може бути другорядною завданням.

Класифікація стандартних серверів

Як правило, кожен сервер обслуговує один (або кілька схожих) протоколів і сервери можна класифікувати за типом послуг, які вони надають.

Універсальні сервери - особливий вид серверної програми, який не надає жодних послуг самостійно. Замість цього універсальні сервери надають серверів послуг спрощений інтерфейс до ресурсів між процесами взаємодії і / або уніфікований доступ клієнтів до різноманітних послуг. Існують кілька видів таких серверів:

Inetd від англ. internet super-server daemon демон сервісів IP - стандартна програма UNIX-систем - програма, що дозволяє писати сервери TCP / IP (і мережевих протоколів інших родин), що працюють з клієнтом через переслані inetd потоки стандартного вводу і виводу (stdin і stdout).

RPC - система інтеграції серверів у вигляді процедур доступних для виклику віддаленим користувачем через уніфікований інтерфейс. Інтерфейс винайдений Sun Microsystems для своєї операційної системи (SunOS, Solaris;

Unix-система), в даний час іспользуетстся як в більшості Unix-систем, так і в Windows.

Прикладні клієнт-серверні технології Windows:

(D-) COM - дозволяє одним програмами виконувати операції над об'єктами даних використовуючи процедури інших програм. Спочатку дана технологія призначена для їх «впровадження і зв'язування об'єктів» (OLE англ. Object Linking and Embedding), але, в загальному, дозволяє писати широкий спектр різних прикладних серверів. COM працює тільки в межах одного комп'ютера, DCOM доступна віддалено через RPC.

Active-X - Розширення COM і DCOM для створення мультимедіа-додатків.

Універсальні сервери часто використовуються для написання всіляких інформаційних серверів, серверів, яким не потрібна якась специфічна робота з мережею, серверів, які не мають ніяких завдань, окрім обслуговування клієнтів. Наприклад, в ролі серверів для inetd можуть виступати звичайні консольні програми та скрипти.

Більшість внутрішніх і мережевих специфічних серверів Windows працюють через універсальні сервери (RPC, (D-) COM).

Мережеві служби забезпечують функціонування мережі, наприклад сервери DHCP і BOOTP забезпечують стартову ініціалізацію серверів і робочих станцій, DNS - трансляцію імен в адреси і навпаки.

Сервери тунелювання (наприклад, різні VPN-сервери) і проксі-сервери забезпечують зв'язок з мережею, недоступною роутингом.

Сервери AAA і Radius забезпечують в мережі єдину аутентифікацію, авторизацію і ведення логів доступу.

Інформаційні служби. До інформаційних служб можна віднести як найпростіші сервери що повідомляють інформацію про хості (time, daytime, motd), користувачів (finger, ident), так і сервери для моніторингу, наприклад SNMP. Більшість інформаційних служб працюють через універсальні сервери.

Особливим видом інформаційних служб є сервери синхронізації часу - NTP крім інформуванні клієнта про точний час NTP-сервер періодично опитує кілька інших серверів на предмет корекції власного часу. Крім корекції часу аналізується і коректується швидкість ходу внутрішнього годинника. Корекція часу здійснюється прискоренням або уповільненням ходу внутрішнього годинника (в залежності від напрямку корекції), щоб уникнути можливих проблем при звичайну перестановку часу.

Файл-сервери є сервери для забезпечення доступу до файлів на диску сервера.

Перш за все, це сервери передачі файлів на замовлення, по протоколах FTP, TFTP, SFTP і HTTP. Протокол HTTP орієнтований на передачу текстових файлів, але сервери можуть віддавати в якості запитаних файлів і довільні дані, наприклад, динамічно створені веб-сторінки, картинки, музику і т. П.

Інші сервери дозволяють монтувати дискові розділи сервера в дисковий простір клієнта і повноцінно працювати з файлами на них. Це дозволяють сервери протоколів NFS і SMB. Сервери NFS і SMB працюють через інтерфейс RPC.

Недоліки файл-серверної системи:

Дуже велике навантаження на мережу, підвищені вимоги до пропускної здатності. На практиці це робить практично неможливою одночасну роботу великої кількості користувачів з великими обсягами даних.

Обробка даних здійснюється на комп'ютері користувачів. Це тягне підвищені вимоги до апаратного забезпечення кожного користувача. Чим більше користувачів, тим більше грошей доведеться витратити на оснащення їх комп'ютерів.

Блокування даних при редагуванні одним користувачем робить неможливою роботу з цими даними інших користувачів.

Безпека. Для забезпечення можливості роботи з такою системою Вам буде необхідно дати кожному користувачеві повний доступ до цілого файлу, в якому його може цікавити тільки одне поле.

Сервери доступу до даних обслуговують базу даних і віддають дані на запит. Один з найпростіших серверів подібного типу - LDAP.

Для доступу до серверів баз даних єдиного протоколу не існує, проте всі сервери баз даних об'єднує використання єдиних правил формування запитів - мова SQL.

Служби обміну повідомленнями дозволяють користувачеві передавати і отримувати повідомлення (зазвичай - текстові).

В першу чергу це сервери електронної пошти, що працюють по протоколу SMTP. SMTP-сервер приймає повідомлення і доставляє його в локальний поштову скриньку користувача або на інший SMTP-сервер (сервер призначення або проміжний). На багатокористувацьких комп'ютерах, користувачі працюють з поштою прямо на терміналі (або веб-інтерфейсі). Для роботи з поштою на персональному комп'ютері, пошта забирається з поштової скриньки через сервери, що працюють по протоколах POP3 або IMAP.

Для організації конференцій існує сервери новин, що працюють по протоколу NNTP.

Для обміну повідомленнями в реальному часі існують сервери чатів, стандартний чат-сервер працює по протоколу IRC - розподілений чат для інтернету. Існує велика кількість інших чат-протоколів, наприклад ICQ або Jabber.

Сервери віддаленого доступу

Сервери віддаленого доступу, через відповідну клієнтську програму, забезпечують користувача консольним доступом до віддаленої системи.

Для забезпечення доступу до командного рядка служать сервери telnet, RSH, SSH.

Графічний інтерфейс для Unix-систем - X Window System, має вбудований сервер віддаленого доступу, так як з такою можливістю розроблявся спочатку. Іноді можливість віддаленого доступу до інтерфейсу X-Window неправильно називають «X-Server» (цим терміном в X-Window називається відеодрайвер).

Стандартний сервер віддаленого доступу до графічного інтерфейсу Microsoft Windows називається термінальний сервер.

Деяку різновид управління (точніше моніторингу і конфігурації), також, надає протокол SNMP. Комп'ютер або апаратний пристрій для цього повинно мати SNMP-сервер.

Ігрові сервери, служать для одночасної гри декількох користувачів в єдиній ігрової ситуації. Деякі ігри мають сервер в основний постачання і дозволяють запускати його у невиділеному режимі (тобто дозволяють грати на машині, на якій запущений сервер).

Серверні рішення - операційні системи та / або пакети програм, оптимізовані під виконання комп'ютером функцій сервера і / або містять в своєму складі комплект програм для реалізації типового сервісів.

Прикладом серверних рішень можна привести Unix-системи, спочатку призначені для реалізації серверної інфраструктури, або серверні модифікації платформи Microsoft Windows.

Також необхідно виділити пакети серверів і супутніх програм (наприклад, комплект веб-сервер JavaScript MySQL для швидкої розгортки хостингу) для установки під Windows (для Unix властива модульна або «пакетна» установка кожного компонента, тому такі рішення рідкісні).

В інтегрованих серверних рішеннях установка всіх компонентів виконується одноразово, всі компоненти в тій чи іншій мірі тісно інтегровані і попередньо налаштовані один на одного. Однак в цьому випадку, заміна одного з серверів або вторинних додатків (якщо їх можливості не задовольняють потреб) може представляти проблему.

Серверні рішення служать для спрощення організації базової IT-інфраструктури компаній, тобто для оперативної побудови повноцінної мережі в компанії, в тому числі і «з нуля». Компонування окремих серверних додатків в рішення має на увазі, що рішення призначене для виконання більшості типових завдань; при цьому значно знижується складність

розгортання і загальна вартість володіння IT-інфраструктурою, побудованою на таких рішеннях.

Проксі-сервер служба в комп'ютерних мережах, що дозволяє клієнтам виконувати непрямі запити до інших мережних служб. Спочатку клієнт підключається до проксі-сервера і запитує який-небудь ресурс (наприклад, e-mail), розташований на іншому сервері. Потім проксі-сервер або підключається до вказаного серверу і отримує ресурс у нього, або повертає ресурс із власного кеша (у випадках, якщо проксі має свій кеш). У деяких випадках запит клієнта або відповідь сервера може бути змінений проксі-сервером в певних цілях. Також проксі-сервер дозволяє захищати клієнтський комп'ютер від деяких мережевих атак.

1.5 Висновки до розділу 1

В даному розділі розглянуто та проаналізовано ринок Android-додатків можливих конкурентів, як було вищесказано. Існують аналоги Не доцільним та не зовсім комфортно, якщо потрібно відстежити декілька точок, то це неможливо.

Також сформовано вимоги до програмного трюку, розглянуто можливі функції для зручності користувачів.

Розглянуто можливі архітектури програмного забезпечення. Освоєно основні цілі та завдання сучасних програмних продуктів. Перелічино технології розробки, обрано можливу структуру проекту, переваги та недоліки даної структури.

Також аналізовано технології розробки та фреймворки для подальшої розробки інформаційної системи.

2 АЛГОРИТМІЧНІ ТА МАТЕМАТИЧНІ МОДЕЛІ ТРЕКІНГУ ОСОБИСТОГО ПРОГРЕСУ В СКЕЙТБОРДИНГУ

2.1 Аналіз та виділення стадій розробки

Програмування - це акт інструктування комп'ютерів для виконання завдань». Ще його називають розробкою або кодінгом .

ПЗ являє собою послідовність інструкцій, які виконуються ПК. Комп'ютер же - це будь-який пристрій, здатне обробляти код.

Під технологією розробки ПЗ розуміють оптимальний спосіб ведення розробки, який за певних умов забезпечить отримання кінцевого трюку з наперед заданими властивостями.

В ході дослідження був структурований існуючий матеріал і на його основі створена технологія розробки клієнт-серверних додатків.

Розробка здійснюється за наступним алгоритмом:

1. Вибір архітектури клієнт-сервера (технічна частина). На початку розробки необхідно вибрати між дворівневою і багаторівневої архітектурою. Вибір в даному випадку залежить від кількості користувачів, які будуть працювати в даній системі, вартості обладнання та подальше обслуговування системи (наприклад, доробка функціоналу).

Детальний опис цих архітектур описані в першому розділі.

2. Вибір мови програмування (програмна частина). Клієнт-сервер можна написати на самих різних мовах. У різних мов присутні свої особливості в реалізації мережевих завдань. Нижче наведено (рисунок 2.1) список десяти мов програмування за індексом TIOBE станом на січень 2020 року ("TIOBE programming community index"- індекс, що оцінює популярність мов програмування, на основі підрахунку результатів пошукових запитів [12]. Для формування індексу використовується пошук в декількох найбільш відвідуваних (за даними Alexa) порталах: Google, Blogger, Wikipedia, YouTube, Baidu, Yahoo !, Bing, Amazon.

Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	^	C#	5.349%	+2.07%
6	5	v	Visual Basic .NET	5.287%	-1.17%
7	6	v	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%
9	15	^^	Swift	1.795%	+0.61%
10	9	v	SQL	1.504%	-0.77%

Рисунок 2.1 – Рейтинг мов програмування

Варто враховувати, що Веб-програмування - це окремий випадок програмування клієнт-серверного додатка. Клієнтом в веб-додатку виступає браузер, а сервером - веб-сервер що дає ряд плюсів:

– Низька вартість впровадження;

– Дуже проста підтримка;

– Незалежність від Операційної системи;

– Доступність з будь-якої точки світу.

У веб-програмуванні зазвичай використовують JavaScript на стороні клієнта, а на стороні сервера можуть бути як один або кілька мов програмування: PHP, Java, C ++, C, JavaScript, ASP.NET, Python і інші.

3. Створення робочого прототипу.

Процес створення прототипу зазвичай складається з кроків:

1. Визначення початкових вимог;

2. Розробка першого варіанту прототипу, який містить тільки призначений для користувача інтерфейс системи;

3. Вивчення прототипу замовником і кінцевими користувачами, отримання зворотного зв'язку про необхідні зміни та доповнення;

4. Переробка та поліпшення прототипу: з урахуванням отриманих зауважень і пропозицій змінюються як специфікації, так і прототип, після цього кроки 3 і 4 можуть повторюватися.

Прототипи дають можливість глибше вникнути в проблему і вжити всіх необхідні проектні рішення ще на ранніх етапах проектування.

4. Тестування. Для клієнт-серверних додатків тестування можна умовно розділити на два рівні:

– Серверна;

– Клієнтська.

На першому (серверному) рівні, тестується взаємодія випускається програмного забезпечення з оточенням, в яке воно буде встановлено:

1. Апаратні засоби (тип і кількість процесорів, обсяг пам'яті, характеристики мережі / мережевих адаптерів і т.д.);

2. Програмні засоби (ОС, драйвера і бібліотеки, стороннє ПЗ, впливає на роботу програми і т.д.);

3. Основний упор тут робиться на тестування з метою визначення оптимальної конфігурації обладнання, що задовольняє необхідним характеристикам якості (ефективність, портативність, зручність супроводу, надійність).

На клієнтському рівні, програмне забезпечення тестується з позиції його кінцевого користувача і конфігурації його робочої станції. На цьому етапі будуть протестовані наступні характеристики: зручність використання, функціональність. Для цього необхідно буде провести ряд тестів з різними конфігураціями робочих станцій:

1. Тип, версія і бітность операційної системи (подібний вид тестування називається кроссплатформне тестування);

2. Тип і версія Android, в разі якщо тестується Android-додаток (Подібний вид тестування називається крос-платформерне тестування);

3. Тип і модель відео адаптера (при тестуванні ігор це дуже важливо);

4. Робота програми при різних дозволах екрану;

5. Версії драйверів, бібліотек і т.д. (Для JAVA додатків версія JAVA машини дуже важлива, теж можна сказати і для .NET додатків щодо версії .NET бібліотеки);

Вже на початковому етапі стає очевидно, що чим більше вимог до роботи програми при різних конфігураціях робочих станцій, тим більше тестів нам необхідно буде провести. У зв'язку з цим, рекомендуємо, по можливості, автоматизувати цей процес, так як саме при конфігураційному тестуванні автоматизація реально допомагає заощадити час і ресурси. Звичайно ж автоматизоване тестування не є панацеєю, але в даному випадку воно виявиться дуже ефективним помічником.

5. Реліз і подальша підтримка. Стабільна версія програми, готова до використання за її призначенням, що пройшла тестування, в яких виправлені основні помилки, але існує ймовірність появи нових, раніше не помічених, помилок.

Для створення клієнт-серверного додатка була обрана дворівнева архітектура і об'єктно-орієнтована мова програмування - Java, для написання всієї програми. Вся обробка даних відбувається на сервері. Дворівневу архітектуру легко можна реалізувати і не вимагає яких-небудь фінансових вкладень.

Також потрібно розробити парсер. Спеціально для можливості перепарсінга сторінок в ретроспективі, для вилучення нових даних в архіві і виправлення помилок реалізовано поділ процесу на вилучення вмісту і власне парсинг сторінок.

Автоматизація процесу парсинга увазі асинхронну постановку завдань Воркер, розподіленим в кластері серверів. Автоматична оцінка черзі асинхроннних Воркер дозволяє гнучко налаштовувати періодичність парсинга, для виключення надмірного накопичення черги через проблеми в мережі або перебоях в роботі сайту-джерела.

Система намагається економити ресурси. При отриманні вмісту сторінки, необхідного для парсинга різних типів об'єктів, запускається процес парсинга однієї і тієї ж сторінки, без повторного отримання вмісту. HTTP-сесія, відкрита для отримання даних, по завершенню виконання завдання закривається і упаковується для наступного завдання. Це дозволяє економити на розгадки капчі.

Структурна модель будувалася так що навіть після релізу можна відносно легко додавати новий функціонал.

2.2 Алгоритмічне та матиматечне рішення планування серверу

Залежно від призначення існує кілька визначень поняття сервер.

1. Сервер (мережа) - логічний або фізичний вузол мережі, що обслуговує запити до одного адресою і / або доменному імені (суміжним доменних імен), що складається з одного або системи апаратних серверів, на якому виконуються один або система серверних програм

2. Сервер (програмне забезпечення) - програмне забезпечення, що приймає запити від клієнтів (в архітектурі клієнт-сервер).

3. Сервер (апаратне забезпечення) - комп'ютер (або спеціальне комп'ютерне обладнання) виділений і / або спеціалізований для виконання певних сервісних функцій.

3. Сервер в інформаційних технологіях - програмний компонент обчислювальної системи, що виконує сервісні функції по запиту клієнта, надаючи йому доступ до певних ресурсів.

В даному випадку представлено програмне забезпечення на архітектурі клієнт-сервер, а саме:

– Клієнт відправляє дані браузеру, в даному випадку клієнт це Android-додаток;

– Сервер отримує дані, створює пакет, перевіряє дані, запаковує пакет та посилає клієнту;

– Клієнт отримує дані, розппаковує пакет та обробляє дані запаковує пакет з новими даними та відправляє серверу на перевірку;

– Сервер отримує пакет, розпаковує обробляє та перепаковує дані та відправляє клієнту.

Діаграма представлена на рисунку 2.2.

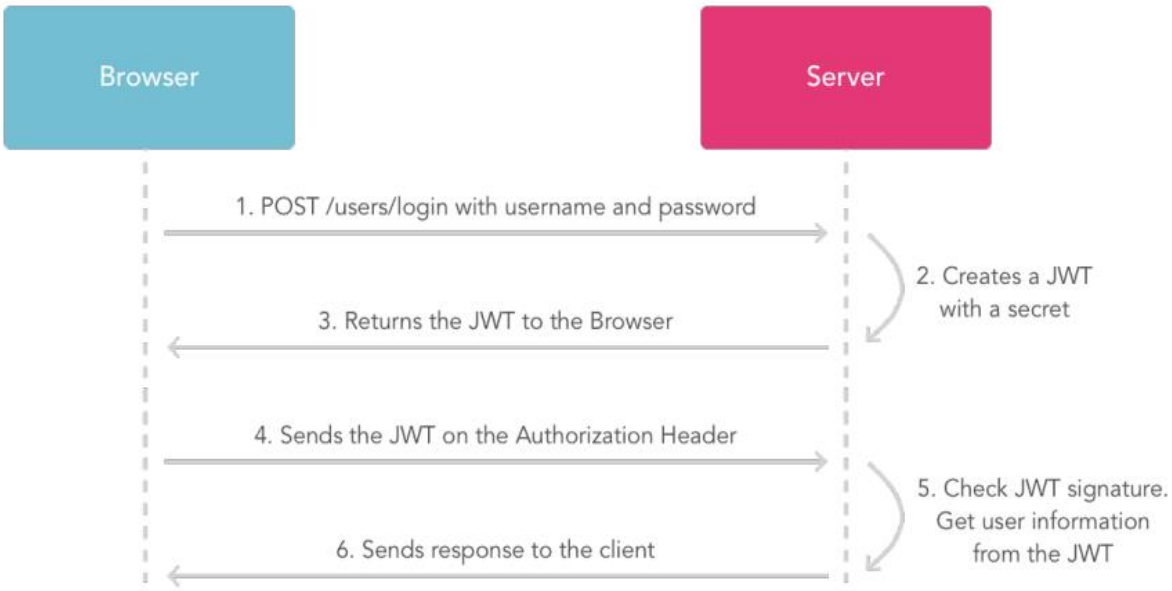


Рисунок 2.2 – Діаграма функціонування клієнта та сервера

Представлено концептуальну модель, отже функціонування інформаційної системи звикористанням клієнт-серверної архітектури.

Основна особливість даної архітектури – фізично розділені два програмні забезпечення, одна яка відповідає за зберігання та обробку даних та клієнт котрий отримує ці дані. Таке розділення дозволяє оптимізувати нагрузку як на мережу так і на апаратне забезпечення звівши всі ризики до мінімуму.

З програмної позиції дану архітектуру реалізовують:

Сервер зберігання;

Сервер обробки;

Клієнт.

Якщо правильно розставити приорітети можна розглянути концептуальну модель на рисунку 2.3.

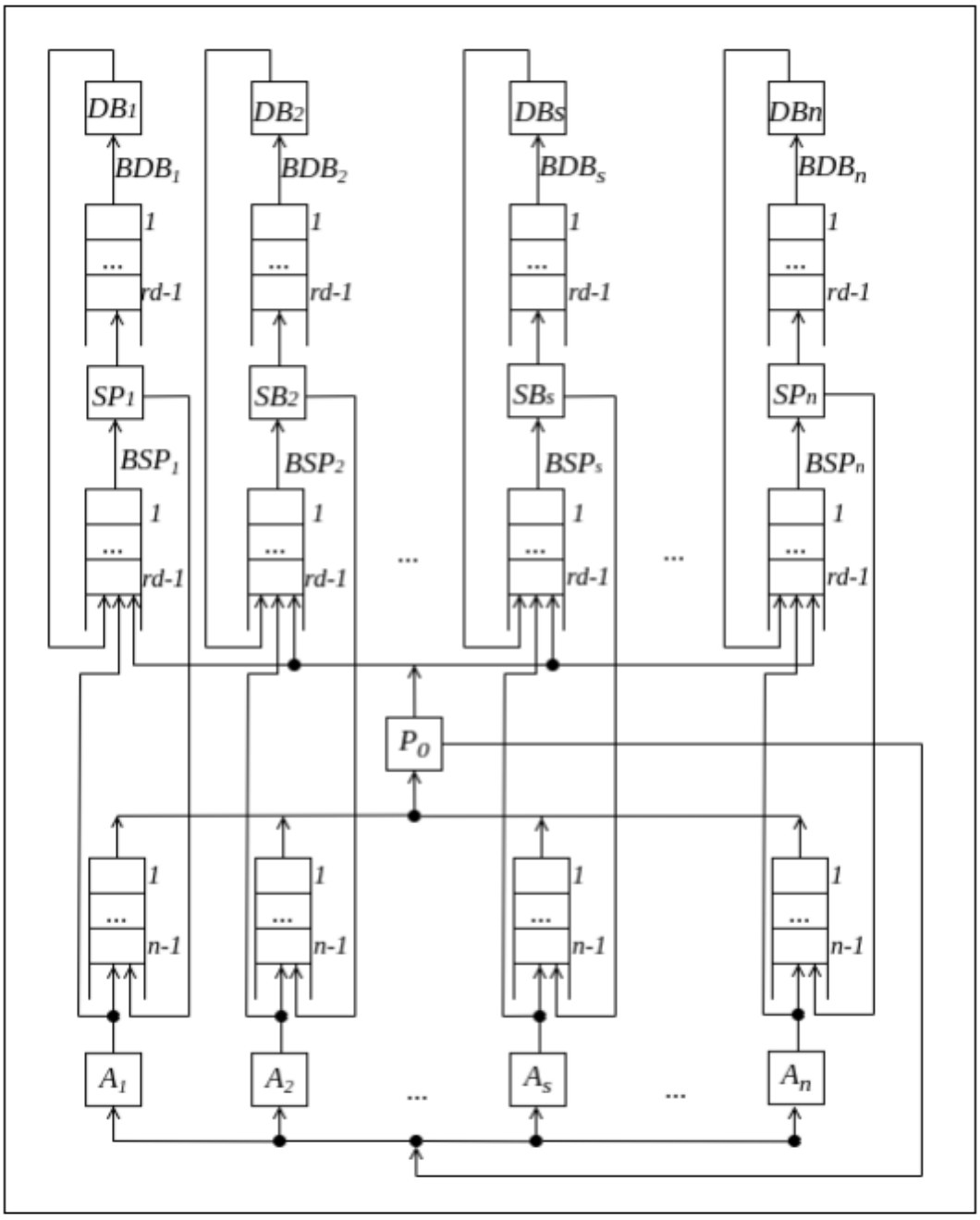


Рисунок 2.3 – Концептуальна схема обміну даних системи

Отже можна розглянути наступним чином як концептуальну модель:

$DB_1, \dots, DB_s, \dots, DB_n,$ - потоки сервра з база даних, які моделюють багато поточність;

$BDB_1, \dots, BDB_s, \dots, BDB_n,$ - буфери пам'яті серверу, які відповідають за дані;

$SP_1, \dots, SP_s, \dots, SP_n,$ - так званні тонкі клієнти, які моделюють роботу клієнт додатків;

$SP_1, \dots, SP_s, \dots, SP_n,$ - буфери пам'яті клієнтів, які відповідають за отримання даних від клієнтів.

Якщо дану архітектуру розглядати як математичну модель то отримаємо потоки даних від клієнтів, які потрібно буде обробляти. Для ідентифікації статусу мережі введено векторний простір , який виглядатиме наступним чином (ф. 2.1):

$$\bar{E} = \left\{ E_i \left(\begin{array}{c} i_{11}, \dots, i_{1s}, \dots, i_{1n}; i_{21}, \dots, i_{2s}, \dots, i_{2n}; \dots \\ \dots; i_{s+1,1}, \dots, i_{s+1,n}, \dots, i_{n+1,1}; \dots \\ \dots; i_{n+1,s}, \dots, i_{n+1,n}, \dots, i_{n+2,1}, \dots \\ \dots, i_{n+2,s}, \dots, i_{n+2,s}, \dots, i_{n+2,n}; \dots \\ \dots; i_{n+s+1,1}, \dots, i_{n+s+1,s}, \dots, i_{n+s+1,n}; \dots \\ \dots; i_{2n+1,1}, \dots, i_{2n+1,s}, \dots, i_{2n+1,n}; \dots \\ \dots; i_{2n+2,1}, \dots, i_{2n+s+1,s}, \dots, i_{2n+2,n}; \dots \\ \dots; i_{2n+s+1,1}, \dots, i_{2n+s+1,s}, \dots, i_{2n+s+1,n}; \dots \\ \dots; i_{3n+1,1}, \dots, i_{3n+1,s}, \dots, i_{3n+1,n} \end{array} \right) \right\}, i = \overline{1,s}$$

2.1

Де:

$$i_s = \begin{cases} 1, \text{якщо } s - \text{й користувач (тонкий клієнт}, s = \overline{1,n}) \\ \text{знаходиться в активному статусі} \\ \text{тобто формує запит на ініціалізацію даних додатку}), \\ 0 - \text{в протилежному випадку}; \end{cases}$$

($i_{sr}, s = \overline{2, n+1}, r = \overline{1,n}$) – описує чергу до каналу і статус каналу, де i_{sr} – кількість запитів r-го користувача (тонкого клієнту) в s-й буфера пам'яті каналу і на обслуговування в каналі;

($i_{sr}, s = \overline{n+2, 2n+1}, r = \overline{1,n}$) – описує чергу до серверів додатків і статуси серверів додатків, де i_{sr} – кількість запитів r-го користувача в буферній пам'яті s-й сервера додатку і на обслуговування в s-им сервером додатку;

($i_{sr}, s = \overline{2n+2, 3n+1}, s = \overline{1, n}$) – описує черги до серверів з базами даних та статус серверів баз даних, де i_{sr} – кількість запитів r-го користувача в буферній пам’яті s-й сервера додатку і на обслуговування в s-им сервером додатку;

При цьому є деякі обмеження.

Представлення інтересів характеристики відображають ймовірності статусу мереж. Нехай стаціонарна ймовірність того що вже знаходиться в статусі де процес потоку змінюється статусом мережі та описується поток представленим вище.

2.3 Алгоритмічне та матиматичне рішення парсера

Дієслово "to parse" в дослівному перекладі означає нічого поганого. Робити граматичний розбір або структурувати - дії корисні і потрібні. Мовою всіх, хто працює з даними на сайтах це слово має свій відтінок.

Парсити - збирати і систематизувати інформацію, розміщену на певних сайтах, за допомогою спеціальних програм, що автоматизують процес.

Якщо ви коли-небудь задавалися питанням, що таке парсер сайту, то ось він відповідь. Це програмні продукти, основною функцією яких є отримання необхідних даних, які відповідають заданим параметрам.

Чи законно використовувати парсинг?

Після з'ясування що таке парсинг, може здатися, що це щось, що не відповідає нормам чинного законодавства. Насправді це не так. Законом не переслідується парсинг. Зате заборонені:

– злом сайту (тобто отримання даних особистих кабінетів користувачів і т. п.);

– DDOS- атаки (якщо на сайт в результаті парсинга даних лягає занадто високе навантаження);

– запозичення авторського контенту (фотографії з копірайтами, унікальні тексти, справжність яких засвідчена у нотаріуса і т. п. краще залишити на їх законному місці).

– Парсинг законний, якщо він стосується збору інформації, що знаходиться у відкритому доступі. Тобто все, що можна і так зібрати вручну.

Парсери просто дозволяють прискорити процес і уникнути помилок через людський фактор. Тому «незаконність» в процес вони не додають.

Інша справа, як власник свіжозібраної бази розпорядиться подібною інформацією. Відповідальність може наступити саме за наступні дії.

Переходимо до того, навіщо ж це може знадобитися. Тут відкривається широкий простір для дій.

Основна проблема сучасного Інтернету - надлишок інформації, яку людина не в змозі систематизувати вручну.

Парсинг використовується для:

Аналізу цінової політики . Щоб зрозуміти середню вартість тих чи інших трюків на ринку, зручно використовувати дані по конкурентах. Однак якщо це сотні і тисячі позицій, зібрати їх вручну швидко неможливо.

Відстеження змін. Парсинг можна здійснювати на регулярній основі, наприклад, щотижня, виявляючи на що підвищилися ціни в середньому по ринку і які новинки з'явилися у конкурентів.

Наведення порядку на своєму сайті. Так, так теж можна. І навіть потрібно, якщо в інтернет-магазині кілька тисяч трюків. Знайти неіснуючі сторінки, дублі, неповне опис, відсутність певних характеристик або невідповідність даних по складських залишків того, що відображається на сайті. З парсером швидше.

Наповнення карток трюків в інтернет-магазині. Якщо сайт новий, рахунок зазвичай йде навіть не на сотні. Вручну на це піде недозволено кількість часу. Часто використовують парсинг з іноземних сайтів, переводять отримані тексти автоматизованим методом, після чого отримують практично готові опису. Іноді те долають з російськомовними сайтами, а отримані тексти

змінюють за допомогою синонімайзер, але за це можна отримати санкції від пошукових систем.

Отримання баз потенційних клієнтів. Існує парсинг, пов'язаний зі складанням, наприклад, списку осіб, які приймають рішення, в тій чи іншій галузі і місті. Для цього може застосовуватися особистий кабінет на сайтах пошуку роботи з доступом до актуальних і архівних резюме. Етичність подальшого використання подібної бази кожна компанія визначає самостійно.

процес перетворення вихідного коду в структурований вигляд. Типовий парсер є комбінацією Лексера і парсера. Лексер групує символи вихідного коду в значущі послідовності, які називаються лексемами. Після цього визначається тип лексеми (ідентифікатор, число, рядок і т.п.). Токеном називається сукупність значення лексеми і її типу. У прикладі на малюнку нижче токенами є sp , = , 100 . Парсер ж з потоку токенов будує зв'язну деревоподібну структуру, яка називається деревом розбору. В даному випадку assign є одним з вузлів дерева. Абстрактне синтаксичне дерево або AST-дерево розбору на більш високому рівні, з якого не видаляються значущі маркери, такі як дужки, коми. Однак існують парсери, в яких крок лексірованія і розбору суміщені як продемонстровано на рис. 2.4.

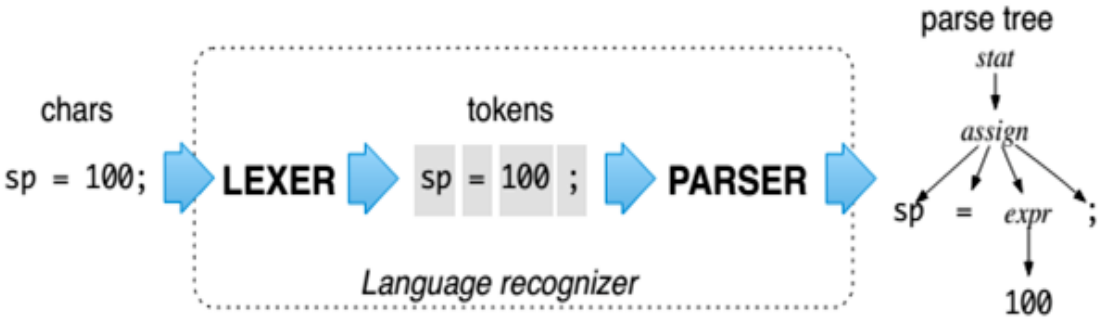


Рисунок 2.4 – Алгоритм дії парсера

Для опису різних вузлів AST використовуються правила. Об'єднання всіх правил називають граматикою мови. Перевагами ручного підходу є те, що парсери, як правило, виходять більш продуктивними і читабельними.

Поставлена математична модель розглянута наступним чином у вигляді векторів даних, які формуються:

$BSP = (BSP_1, \dots, BSP_s, \dots, BSP_n)$, – кількість потоків з'єднання сервера з парсером;

$q = (q_1, \dots, q, \dots, q_n)$, – кількість запитів які формуються парсером;

$rd = (rd_1, \dots, rd_2, \dots, rd_n)$, – кількість інформації яка передається парсером серверу для формування каталогів;

$n = (n_1, \dots, n_j)$, – формування каналів підключення даних між парсером та сервером;

P – сформований запит.

Взаємовідношення векторів у загальні структурі представлено на вищесказаній схемі на рисунку 2.3.

2.4 Алгоритмічне та математичне рішення клієнта

Перш ніж почати написання додатка для Android потрібно зрозуміти цілі, покладені на проект. Які потреби користувачів він повинен задовольняти? Яка

його цінність для цільової аудиторії? Відповівши на ці питання, можна чітко зрозуміти, який саме продукт потрібний. Саме тому створення додатка для Android має включати в себе попередній аналіз ринку клієнта, активності конкурентів та переваг цільової аудиторії.

Створення Android-додатка відрізняється від розробки сайту, але і тут важливо дотримуватись законів розробки. Адже клієнти будуть їх використовувати з певними цілями, а це означає, що важливо зробити додаток інтуїтивно зрозумілим, його структуру – простою і оформлення – приємним для очей. Щоб хотілось використовувати його щодня. Адже багато хто, ледь установка додатків Android здійснена, забувають про існування додатку, якщо додаток не корисний. Розроблено прототип ключових розділів додатку, після чого створено макети. Розробка мобільних додатків під Android, в тому числі

і етап дизайну, – складний процес, який вимагає специфічних знань і глибокого розуміння особливостей розробки та принципів дизайну. Дизайн додатку готовий тепер потрібно зробити так, щоб усе, що намальовано і придумано, працювало. Тут у справу вступають впровадження технологій, що є одним з найважливіших етапів, які включає розробка додатків для Android.

Аналізувавши багато додатків для смартфонів було вирішено додати п’ять активностей, які дозволяють переходити між екранами і взаємодіяти між собою.

Виділено такі активності:

- RegisterActivity;
- ListActivity;
- MyListActivity;
- MyFavoriteListActivity;
- MyProfileActivity.

Навігація по розділах мобільного додатка коли користувач відкриває програму, йому повинно бути легко і швидко знайти необхідний контент. Це головний принцип, на базі якого повинна будуватися вся навігація по розділах. Взаємодію між розділами зображено на рисунку 2.5.

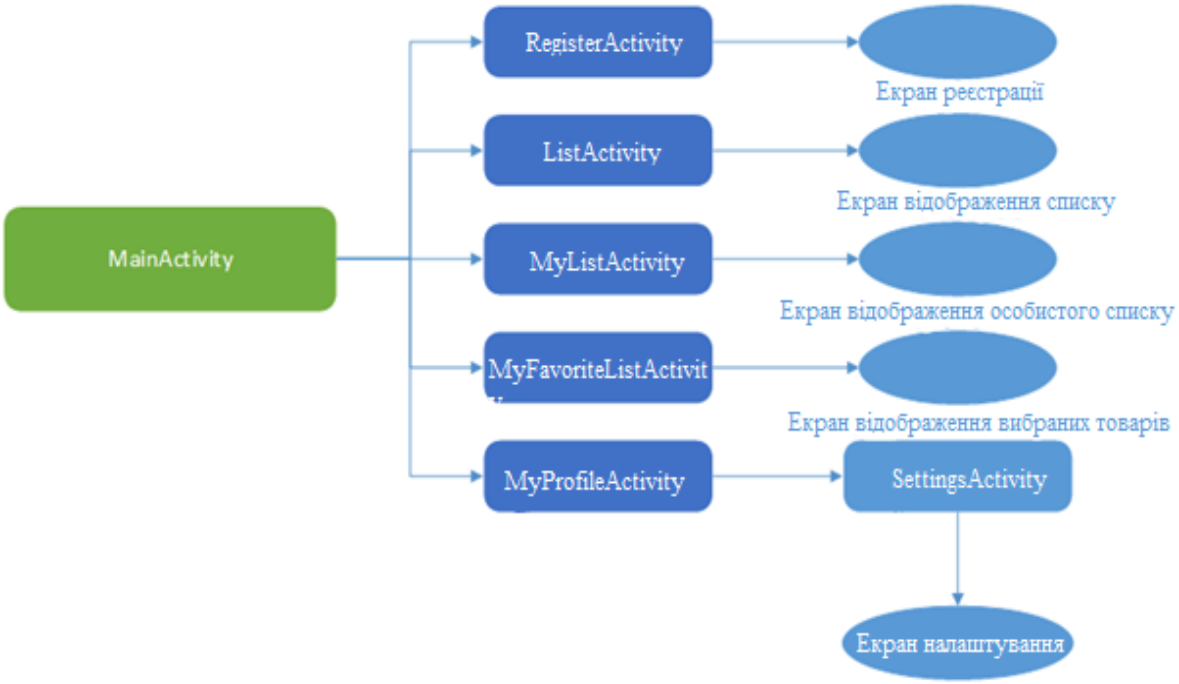


Рисунок 2.5 – Ілюстрація того, як активності утворюють відгалуження у макеті та містять активності

Протягом життя одного з видів Activity, система викликає основний набір методів життєвого циклу в послідовності, аналогічній ступінчастій піраміді. Тобто, кожен етап життєвого циклу Activity є окремим кроком на піраміді. Коли система створює новий екземпляр Activity, кожен метод зворотнього виклику

переміщує стан Activity на один крок до вершини. Верхня частина піраміди є точкою, в якій Activity виконується на передньому плані, і користувач може взаємодіяти з нею.

Як тільки користувач починає покидати Activity, система викликає інші методи, які переміщають стан Activity донизу піраміди, щоб демонтувати Activity. У деяких випадках, Activity буде рухатися тільки частиною шляху вниз по піраміді і чекати (наприклад, коли користувач перемикається на інший додаток), звідки Activity може повернутися в початок (якщо користувач повертається до Activity) і продовжити, де користувач зупинився (рисунок 2.6).

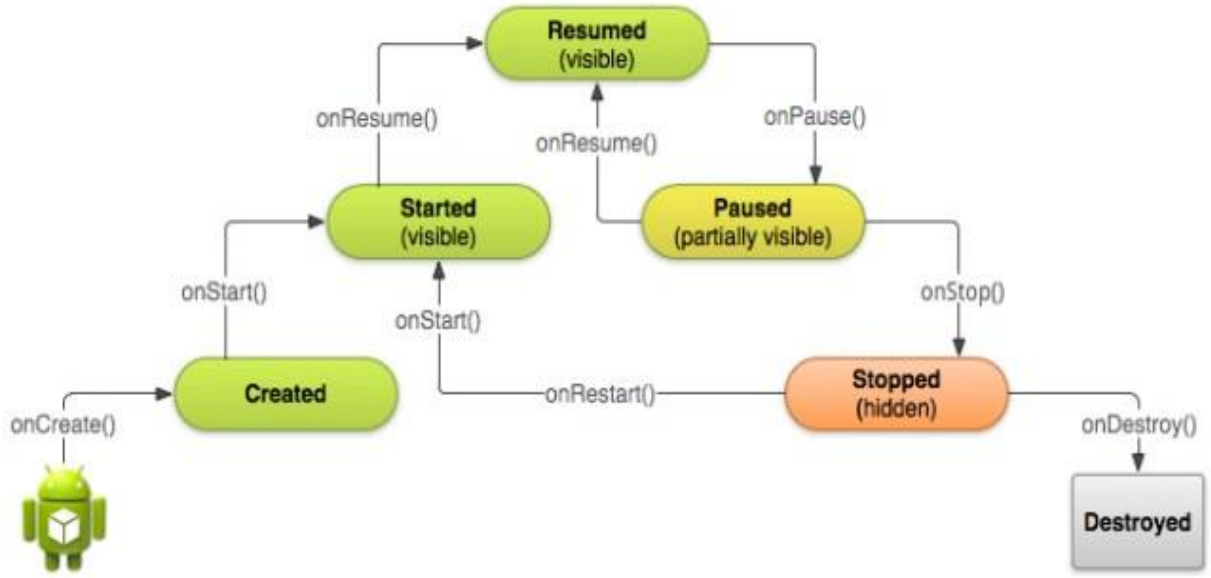


Рисунок 2.6 – Ілюстрація життєвого циклу Activity, виражена у вигляді ступінчастої піраміди.

Малюнок демонструє, як кожний зворотній виклик, використаний, щоб зробити крок Activity до заключного стану на вершині, де метод зворотного виклику змушує Activity зробити крок вниз. Activity може також повернутися до відновленого стану з станів Paused і Stopped.

Залежно від складності Activity, ймовірно, не потрібно реалізовувати всі методи життєвого циклу. Тим не менш, важливо, щоб розуміти кожен стан і реалізовували стани, які забезпечують поведінку додатку таку, яку очікують користувачі. Правильна реалізація методів життєвого циклу Activity гарантує, що додаток поводиться чудово, кількома способами, в тому числі, що він:

– не припиняє свою роботу, якщо користувач отримує телефонний дзвінок або перемикається на інший додаток під час користування вашим додатком;

– не використовує цінні системні ресурси, коли користувач не використовує його активно;

– не втрачається робота користувачів, якщо вони залишають додаток і повертаються до нього пізніше;

– не втрачає роботу користувача, коли екран змінюється між альбомною і портретною орієнтаціями.

Ще є кілька ситуацій, в яких Activity переходить між різними станами, які зображені на рисунку 2.7.

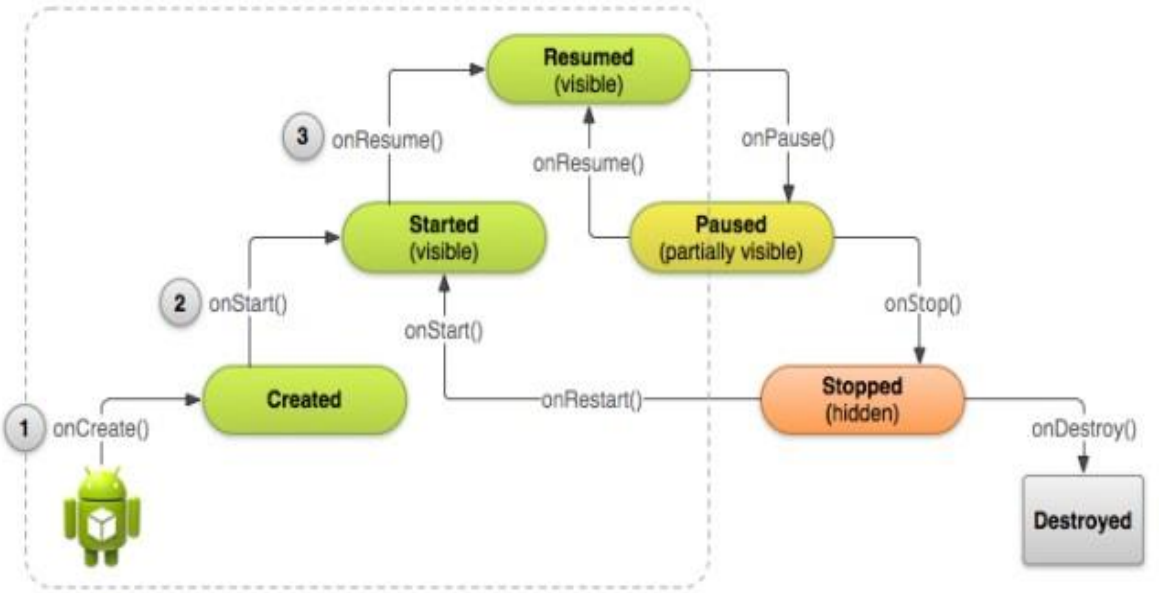


Рисунок 2.7 – Ілюстрація структури життєвого циклу Activity з акцентом на трьох основних зворотних викликах

Однак, тільки три з цих станів можуть бути статичними. Тобто, Activity може існувати тільки в одному з трьох станів протягом тривалого періоду часу:

– resumed (поновлення).У цьому стані Activity знаходиться на передньому плані і користувач може взаємодіяти з нею. (Також іноді називають «запущеним» станом) ;

– paused (призупинено). У цьому стані Activity частково закрита деякою роботою іншої Activity, на передньому плані вона напівпрозора або не покриває весь екран. Призупинена Activity не приймає введення користувача і не може виконати який-небудь код;

– stopped (зупинено). У цьому стані Activity повністю прихована і не видна користувачеві; це ще називають фоновим режимом. Під час зупинки

екземпляр Activity і всі його дані про стан, такі як змінні членів зберігаються, але вона не може виконати який-небудь код.

Інші стани (Created і Started) змінні і система швидко переходить від них до наступного стану, викликавши наступний метод зворотнього виклику життєвого циклу. Тобто, після того, як система викликає onCreate(), вона швидко викликає onStart(), яка потім швидко переходить до onResume().

Коли користувач вибирає піктограму додатку на головному екрані, то система викликає метод onCreate() для Activity. Це Activity, яка служить основною точкою входу в інтерфейс користувача додатку.

Визначено, MainActivity буде використовуватися, як основна Activity, у файлі маніфесту Android, AndroidManifest.xml, який знаходиться в корені каталогу проекту.

Більшість додатків включають в себе кілька різних видів Activity, які дозволяють користувачеві виконувати різні дії тому було створено окремий пакет «games» та діаграму класів для розробки взаємодії класів у пакеті (додаток Д). Коли Activity є основною Activity, то створюється, коли користувач натискає іконку додатку або іншої Activity, яку запускає додаток у відповідь на дії користувача. Система створює кожний новий екземпляр Activity шляхом виклику методу onCreate().

Необхідно реалізувати метод onCreate(), щоб виконати основну логіку запуску додатку, що має відбутися лише один раз за весь термін життя Activity. Наприклад, реалізація onCreate() повинна визначити користувальницький інтерфейс і, можливо, екземпляри деяких змінних класу області видимості.

Наступний метод onCreate() демонструє фундаментальні налаштування для Activity. Такі, як оголошення інтерфейсу користувача (визначений у XML-файлі макета), визначення змінних членів та налаштування деяких UI:

Після завершення виконання onCreate(), система викликає методи onStart() і onResume() у швидкій послідовності. Activity ніколи не перебуває в станах Created або Started. Технічно, Activity стає видимою для користувача, коли викликано onStart(), але негайно слідує onResume() і Activity залишається

у стані Resumed поки щось не відбувається, щоб змінити, наприклад, при отриманні телефонного дзвінка користувач переходить до іншої Activity, або екран пристрою гасне.

Ці виклики система робить послідовно при створенні нового екземпляра Activity: onCreate(), onStart() і onResume(). Після того, як ця послідовність зворотних викликів завершується, Activity досягає стану Resumed, в якому користувачі можуть взаємодіяти з Activity, поки вони не переключитися на іншу Activity.

Знищення Activity, у той час, як перший зворотній виклик життєвого циклу Activity є onCreate(), її самий останнім зворотнім викликом є onDestroy(). Система викликає цей метод на вашій Activity як кінцевий сигнал про те, що екземпляр Activity буде повністю видалений з пам'яті системи.

Більшості додатків не треба реалізувати цей метод, тому що посилання локального класу знищуються з Activity і Activity повинна бути повністю очищена через onPause() і onStop(). Однак, якщо інші тривалі ресурси, які можуть потенційно викликати витік пам'яті, якщо їх не закрити, то потрібно знищити за допомогою onDestroy().

Система викликає onDestroy() після того, як вже викликані onPause() і onStop() у всіх ситуаціях, крім однієї: коли відбувається звернення finish() зсередини методу onCreate(). У деяких випадках, наприклад, коли Activity функціонує як тимчасове рішення розробника, щоб запустити іншу Activity, можна викликати finish() з onCreate(), щоб знищити(видалити) Activity. У цьому випадку система негайно викликає onDestroy() без виклику будь-яких інших методів життєвого циклу.

Макет Списку з розміщенням елементів, які будуть переглядатися та гортатися вверх-вниз відповідно до бажання користувача, також елементи які будуть розміщені, а саме піктограма, назва та опис товару який знаходитиметься в списку продемонстровано на рисунку 2.8.

Рисунок 2.8 – Макет Списку

При розробці меню навігації в додатку, було створено відповідний макет. Виділено п’ять навігаційних блоків(екранів). Для того, щоб захопити користувачів, надано необхідну їм інформацію безпосередньо на першій сторінці із чіткими інструкціями до дії. Користувачам складно орієнтуватися по розділах меню, коли воно не збігається з прийнятими принципами категоризації. Розділи меню повинні бути позначені чітко, їх вміст не повинен співпадати навіть частково.

Поставлена математична модель розглянута наступним чином у вигляді векторів даних, які формуються:

$A = (A_1, \dots, A_2, \dots, A_n)$, – декілька клієнтських додатків (Андроїд-додатки);

$SP = (SP_1, \dots, SP_s, \dots, SP_n)$, – декілька каналів з’єднання з сервером;

$DB = (DB_1, \dots, DB_s, \dots, DB_n)$, – кількість потоків з'єднання з базою даних;

$Q = (Q_1, \dots, Q_2, \dots, Q_n)$, – кількість запитів до бази даних які формуються клієнтом;

$r = (r_1, \dots, r_2, \dots, r_q)$, – можлива кількість клієнтів додатків, які звернуться до сервера;

$V = (V_1, \dots, V_j, \dots, V_d)$, – об'єм відношень даних між клієнтом та сервером;

$VSP = (VSP_1, \dots, VSP_s, \dots, VSP_n)$, – швидкість зчитування даних клієнтом;

$DSP = (DSP_1, \dots, DSP_s, \dots, DSP_n)$, – швидкість запису оброблюваних даних клієнта;

Взаємовідношення представлено на вищесказаній схемі на рисунку 2.3.

2.5 Висновки до розділу 2

В даному розділі представлено математичні моделі та деяке алгоритмічне рішення щодо полегшення розробки інформаційної

Проаналізовано технології обрано відповідні архітектури для системи. Також виділено стадії розробки окремих модулів, а саме:

- Сервер;
- Парсер;
- Клієнт.

Отже сервер відповідає за зберігання та обробку даних, також проаналізовано модель передачі даних та швидке реагування сервера.

Щодо парсера обрано гнучку модель одну з найпопулярніших, де парсер, проганяє по сторінкам та шукає потрібну інформацію, формує та надсилає їх серверу для обробки.

3 АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ РІШЕННЯ ТРЕКІНГУ ОСОБИСТОГО ПРОГРЕСУ В СКЕЙТБОРДИНГУ

3.1 Розробка та функціонування сервера та парсера

Парсинг («синтаксичний аналіз») – завдання розібрати і перетворити в осмислені одиниці щось, написане на деякому фіксованому мовою, будь то мова програмування, мова розмітки, мова структурованих запитів або головний мову життя, Всесвіту і всього такого. Типова послідовність етапів вирішення завдання виглядає приблизно так:

Описати мову . Звичайно, спочатку треба визначитися, яку саме задачу ми вирішуємо. Зазвичай опис мови - це чергова варіація форми Бекуса-Наура . (Ось , наприклад, опис граматики Python, що використовується при побудові його парсеру.) При цьому встановлюються як правила «побудови речень» в мові, так і правила визначення валідних слів.

Розбити введення на токени . Пишеться лексичний аналізатор (в народі токенайзер), який розбиває вхідні рядок або файл на послідовність токенів , тобто валідних слів нашої мови (або ниє, що це не можна зробити).

Перевірити синтаксис і побудувати синтаксичне дерево . Перевіряємо, чи відповідає послідовність токенів опису нашої мови. Тут в хід йдуть алгоритми на кшталт методу рекурсивного спуску . Кожне валідності пропозицію мови включає якесь кінцеве кількість валідних слів або інших валідних пропозицій; якщо токени змогли скластися в стрункy картину, то на виході ми автоматично отримуємо дерево, яке і називається абстрактним синтаксичним деревом .

Написання парсеру проілюструємо на простому, але не до кінця тривіальний прикладі - парсінгу JSON. Граматика виглядає приблизно так:

```
root ::= value
value ::= string | number | object | array | 'true' |
'false' | 'null'

array ::= '[' ']' | '[' comma-separated-values ']'
```

```
comma-separated-values ::= value | value ',' comma-
separated-values

object ::= '{' '}' | '{' comma-separated-keyvalues '}'
comma-separated-keyvalues ::= keyvalue | keyvalue ',' comma-
separated-keyvalues
keyvalue ::= string ':' value
Парс JSON
def parse_number(src):
    match = number_regex.match(src)
    if match is not None:
        number, src = match.groups()
string_regex = re.compile(r"('(?:[^\\']|\\['\\/bfnrt]|\\u[0-
9a-fA-F]{4})*?')\s*(.*)", re.DOTALL)
def parse_string(src):
    match = string_regex.match(src)
    if match is not None:
        string, src = match.groups()
        return eval(string), src
```

Для всього іншого напишемо одну функцію, яка генерує простенькі функції-парсери:

```
def parse_word(word, value=None):
    l = len(word)
    def result(src):
                if src.startswith(word):
            return value, src[l:].lstrip()
    result.__name__ = "parse_%s" % word
    return result

parse_true = parse_word("true", True)
parse_false = parse_word("false", False)
parse_null = parse_word("null", None)
```

Разом, за яким принципом ми будуємо наші функції:

Вони приймають рядок, яку потрібно парсити.

Вони повертають пару при успіху (тобто коли необхідна конструкція знайшлася на початку рядка) і None при провалі.

Власне, на цих трьох функціях проблеми з токенами вирішені

Парс правило з розгалуженням

```
def parse_value(src):
    match = parse_string(src)
    if match is not None:
        return match
    match = parse_number(src)
    if match is not None:
        return match
    # ...
```

Тепер повертаємо генератори - порожні, якщо парсинг не вдався, і рівно з одним елементом, якщо вдався. Функції даному стилі:

```
number_regex = re.compile(r"(-?(?:0|[1-
9]\d*)(?:\.\d+)?(?:[eE][+-]?\d+)?)\s*(.*)", re.DOTALL)

def parse_number(src):
    match = number_regex.match(src)
    if match is not None:
        number, src = match.groups()
        yield eval(number), src

string_regex = re.compile(r"('(?:[^\\']|\\['\\/bfnrt]|\\u[0-
9a-fA-F]{4})*?')\s*(.*)", re.DOTALL)

def parse_string(src):
    match = string_regex.match(src)
    if match is not None:
        string, src = match.groups()
        yield eval(string), src
def parse_word(word, value=None):
    l = len(word)
    def result(src):
        if src.startswith(word):
            yield value, src[l:].rstrip()
    result.__name__ = "parse_%s" % word
    return result
parse_true = parse_word("true", True)
parse_false = parse_word("false", False)
parse_null = parse_word("null", None)
def parse_value(src):
    for match in parse_string(src):
        yield match
        return
    for match in parse_number(src):
        yield match
        return
    # ...
def parse_value(src):
    for match in chain(
        parse_string(src),
        parse_number(src),
        parse_array(src),
        parse_object(src),
        parse_true(src),
        parse_false(src),
        parse_null(src),
    ):
        yield match
        return
```

При цьому ефективність залишається на колишньому рівні - кожна функція почне виконуватися (а отже, робити роботу, перевіряючи регулярні вирази) тільки тоді, коли попередня не дасть результату. Return гарантує, що зайва робота не буде виконана, якщо десь в середині списку парсинг вдався.

Лістинг головних функцій парсера представлено у додатку Г.

3.3 Розробка та функціонування клієнта

Є кілька сценаріїв, в яких Activity руйнується при нормальній поведінці додатків, наприклад, коли користувач натискає кнопку Back або власні сигнали Activity самі її руйнують, викликаючи finish().

Система також може знищити Activity, якщо вона на даний час зупинена і не використовувалася протягом тривалого часу або передній план Activity вимагає більше ресурсів, для чого система повинна закрити фонові процеси для відновлення пам'яті.

Activity знищується, коли користувач натиснув Back або Activity сама закінчує себе, то концепція системи даного екземпляру Activity йде назавжди, оскільки поведінка вказує, що Activity більше не потрібна. Однак, якщо система руйнує Activity через системні обмеження, а не через нормальну поведінку додатка, то хоча фактично екземпляр Activity йде, система запам'ятовує, що така Activity існувала і, якщо користувач переходить назад до неї, то система створює новий екземпляр Activity з використанням набору збереженої інформації, який описував стан Activity, коли вона була зруйнована. Збережені дані, які система використовує для відновлення попереднього стану, називаються "стан екземпляра" і є колекцією пари ключ-значення, що зберігаються в об'єкті Bundle.

Activity буде зруйнована і відтворена щоразу, коли користувач обертає екран. Коли екран змінює орієнтацію, то система знищує і відтворює передній план Activity, тому що конфігурація екрану змінилася і вашій Activity, можливо, треба буде звантажити альтернативні ресурси.

За замовчуванням, система використовує стан екземпляра Bundle, щоб зберегти інформацію про кожен об'єкт View в макеті Activity. Тому, якщо екземпляр Activity зруйнований і відтворений, то стан компоновки відновлений у своєму колишньому стані без необхідного коду. Тим не менш, Activity може мати більше інформації про стан, якщо відновити, наприклад, змінних членів, які відстежують прогрес користувача в Activity.

Для того, щоб система Android відновила стан вигляду Activity, кожен вигляд повинен мати унікальний ідентифікатор, який додається до атрибута android:id.

Щоб зберегти додаткові дані про стан Activity, необхідно перевизначити метод зворотного виклику onSaveInstanceState(). Система викликає цей метод, коли користувач залишає свою Activity і передає його об'єкту Bundle, який буде збережений в тому випадку, коли Activity припиняє своє функціювання. Якщо система повинна відтворити екземпляр Activity пізніше, то вона проходить той же самий об'єкт Bundle, як до onRestoreInstanceState() так і до onCreate() методів як зображено на рисунку 3.3.

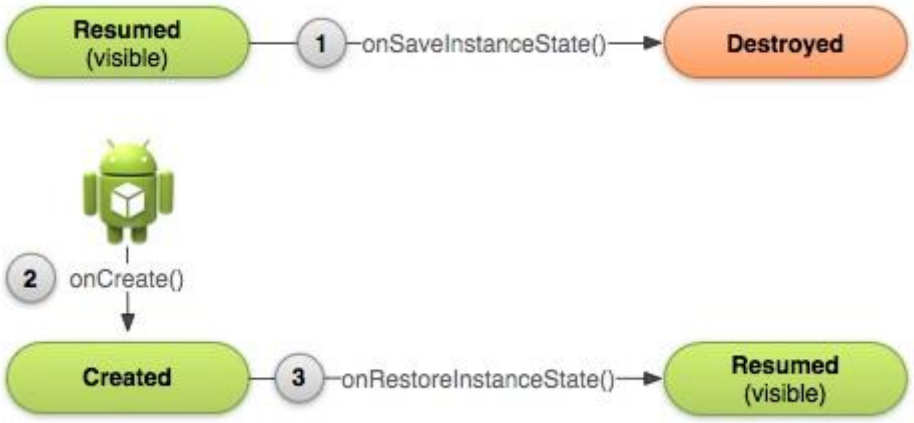


Рисунок 3.3– Стани Activity

Коли система починає зупиняти Activity, вона викликає onSaveInstanceState(), так що можна вказати додаткові дані про стан, який б потрібно зберегти в разі, коли екземпляр Activity повинен бути відтворений.

Якщо Activity буде знищена і має бути відтворений той самий екземпляр, то система передає дані стану, заданого в, як метод onCreate() і метод onRestoreInstanceState().

Збереження стану Activity. Коли Activity починає зупинятися, система викликає onSaveInstanceState(), тому Activity може зберегти інформацію стану з

колекцією пар ключ-значення. Реалізація за замовчуванням цього методу зберігає інформацію про стан вигляду ієрархії Activity, такого, як текст у віджеті EditText або положення прокрутки в ListView.

Щоб зберегти додаткову інформацію про стан для Activity, потрібно реалізувати onSaveInstanceState() і додати пари ключ-значення до об'єкта Bundle:

```
static final String STATE_SCORE = "playerScore";
static final String STATE_LEVEL = "playerLevel";
...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
  // Збережіть нинішній стан гри користувача
  savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
  savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

  // Звиклик superclass, бо це зберігає стан вигляду
ієрархії
  super.onSaveInstanceState(savedInstanceState);
}
```

Повний програмний код наведено в додатку А.

Відновлення стану Activity. Коли Activity відтворена після свого попереднього руйнування, то можна відновити збережений стан з Bundle, що система виконувала свою Activity. Обидва методи зворотніх викликів onCreate() і onRestoreInstanceState() отримують той самий Bundle, що містить інформацію про стан екземпляра було розроблено UML-діаграма класів (додаток В) .

Тому що метод onCreate() викликається, коли система створює новий екземпляр Activity або відтворює попередній, то потрібно перевірити, чи стан Bundle є недійсним, перш ніж намагатися прочитати його. Якщо це null, то

система створює новий екземпляр Activity, замість відновлення попереднього, який був зруйнований. Відновлення даних про стан в onCreate():

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState); // Завжди викликайте
superclass першим

  // перевірка, чи буде відтворення раніше зруйнованого
екземпляру
  if (savedInstanceState != null) {
    // Відновіть значення пам’яті зі збереженого стану
    mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
    mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
  } else {
    // ініціалізація членів зі значеннями за замовчуваннями
для нового екземпляру
  }
}
```

Повний програмний код наведено в додатку А.

Замість того, щоб відновити стан за допомогою onCreate(), потрібно вибрати для реалізації onRestoreInstanceState(), який викликає система після методу onStart(). Система викликає onRestoreInstanceState(), тільки якщо є збережений стан, щоб відновити, тому що не потрібно перевіряти, чи Bundle є Null:

```
public void onRestoreInstanceState(Bundle
savedInstanceState) {
  // викликано superclass, бо так можете відновити вигляд
ієрархії
  super.onRestoreInstanceState(savedInstanceState);

  // Відновлено членів стану зі збереженого екземпляру
  mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
  mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

Повний програмний код наведено в додатку В.

Відштовхуючись від макетів та діаграми варіантів використання (додаток Г) було написано навігаційне меню на мові розмітки XML. Створено файл main_Activity.xmls.

Потрібно було створити п’ять кнопок для переходу у інші активності для цього використано макет LinearLayout представлений двома варіантами - Horizontal і Vertical. Макет LinearLayout вирівнює всі дочірні об'єкти в одному

напрямку - вертикально або горизонтально. Напрямок задається за допомогою атрибута орієнтації android: orientation:

– android: orientation = horizontal;
– android: orientation = vertical.

Всі дочірні елементи поміщаються в стек один за іншим, так що вертикальний список компонентів матиме тільки один дочірній елемент в ряду незалежно від того, наскільки широким він є. Горизонтальне розташування списку буде розміщувати елементи в один рядок з висотою, що дорівнює висоті найвищого дочірнього елемента списку. Код файла розмітки наведено нижче:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="interpreter.pac.MainActivity"
    android:orientation="vertical"
    android:padding="10dp"
    android:background="#fcfcfc"
    android:gravity="center"
    <LinearLayout
        android:clipToPadding="false"
        android:gravity="center"
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <android.support.v7.widget.CardView
            android:id="@+id/lessId"
            android:layout_width="340dp"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            android:clickable="true"

android:foreground="?android:attr/selectableItemBackground">
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:gravity="center"
                android:orientation="vertical">
                <ImageView
                    android:layout_width="64dp"
                    android:layout_height="64dp"
```

```
android:background="@drawable/cerclebackgroundgreen"
                    android:padding="10dp"

android:src="@drawable/ic_control_point_black_24dp" />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="10dp"
                    android:text="@string/less"
                    android:textStyle="bold" />
                <View
                    android:layout_width="match_parent"
                    android:layout_height="1dp"
                    android:layout_margin="10dp"
                    android:background="@color/lightgray" />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:gravity="center"
                    android:padding="5dp"
                    android:text="@string/other_less"

android:textColor="@android:color/darker_gray" />
            </LinearLayout>
        </android.support.v7.widget.CardView>
    </LinearLayout>
</LinearLayout>
```

Також створено файл для форми входу, код розмітки:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent"
    <ImageView
        android:layout_width="150dp"
        android:layout_height="150dp"
        android:src="@drawable/logo"
        android:layout_gravity="center"
        android:layout_marginTop="30dp"/>
    <android.support.design.widget.TextInputLayout
        android:id="@+id/input_log_email"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp">
        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Email"
```

```
            android:inputType="textEmailAddress" />
    </android.support.design.widget.TextInputLayout>
    <android.support.design.widget.TextInputLayout
        android:id="@+id/input_log_pass"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="5dp">
        <android.support.design.widget.TextInputEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="Password"
            android:inputType="textPassword" />
    </android.support.design.widget.TextInputLayout>
    <Button
        android:id="@+id/btn_log"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="10dp"
        android:text="Log in" />
</LinearLayout>
```

Після компіляції додаток необхідно відкрити. Оператора вітатиме екран входу. Є можливість використовувати додаток без входу і реєстрації, але при такій умові, оператору будуть доступний обмежений функціонал, а саме тільки перегляд акцій.

3.4 Тестування системи

Сьогодні є безліч фреймворків для тестування, що підтримують практично всі існуючі мови. Здавалося б - можна брати і автоматизувати. Але навіть зараз ручні тести важливі.

Одне з пояснень їх необхідності полягає в тому в тому, що при ручному тестуванні функціоналу ми можемо набагато швидше отримати інформацію про стан трюку, який аналізуємо, про якість розробки. Крім того, при автоматизації попередньо розроблені кейси часто доводиться міняти і актуалізувати, а на написання Автотест потрібен певний час.

Все це означає, що головна мета ручних тестів - попередньо переконатися в тому, що заявлений функціонал працездатний, не має помилок і видає очікувані, заплановані результати. Без них не можна бути впевненим у тому, що можна працювати далі. Особливо це актуально для функцій, на

реалізацію яких зав'язана подальша розробка. В такому випадку метушня зі створенням Автотест на ці фічі стає блокуючим фактором для всієї розробки трюку, зрушуючи терміни і зриваючи дедлайни. Момент, коли кейси прийде пора автоматизувати, все одно рано чи пізно настане - але не варто прагнути наблизити його штучно в гонитві за тотальним винятком ручної праці.

Це особливо актуально для мобільного розробки. Більшість таких проектів сьогодні розробляється короткими спринту, а значить фичи в них впроваджуються в прискореному темпі. У подібних умовах ручне тестування дозволяє максимально оперативно давати команді зворотний зв'язок: повідомляти про наявність багів - або радувати її тим, що все окей і можна рухатися далі. Провести серію Автотест ви зможете пізніше, покривши з їх допомогою великі масиви коду. Ручне тестування допоможе підготувати кейси для цієї перевірки.

Отже для тестування системи запущено сервер, який передаватиме та отримуватиме запити, на рисунку 3.4 продемонстровано запуск сервера та отримання запитів.

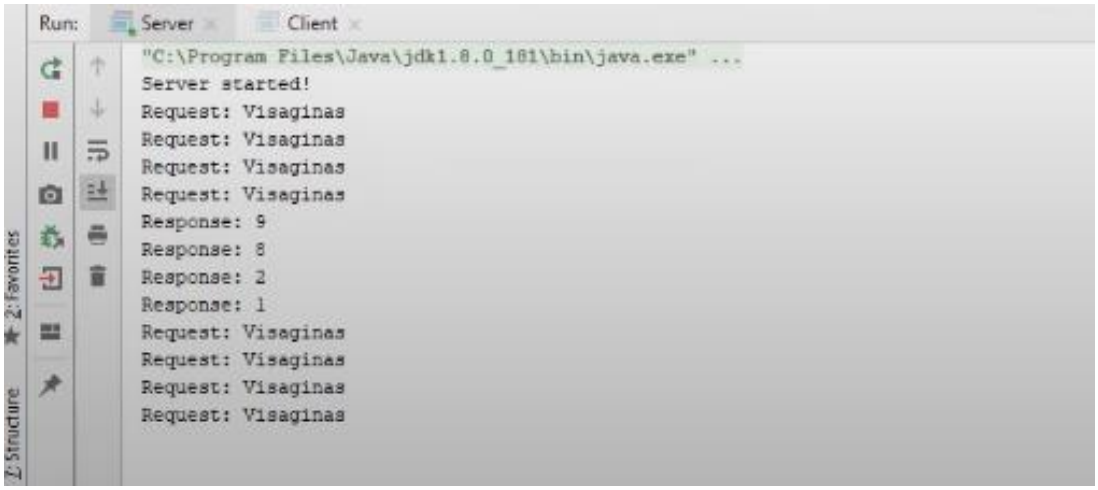


Рисунок 3.4 – Знімок екрану обробки запитів

При вході в додаток нас зустрічає екран з привітанням, а після завантаження можемо побачити перелік магазин доступних розділів, все це можна переглянути на рисунку 3.5 представленому нижче.

Реєстрація дає можливість роботи зі списками трюків, тому для входу потрібно увійти (рис. 3.5). Якщо ж клієнт вже зареєстрований в системі він може увійти, а опісля переглянути списки трюків.

Рисунок 3.5 – Екран реєстрації

Увійшовши в обліковий запис або вибравши режим без реєстрації, користувач потрапляє на головний екран додатка. Екран списку розділів в додатку зображено на рисунку 3.6.

Рисунок 3.6 – Екран списку розділів

На даному знімку екрану зображено (рис. 3.6) зображено список розділів Зручний список «Улюблені» з кнопкою «+» дозволяє користувачу додати в обрані супермаркет. Також далі, з будь-якої активності користувач може викликати розділ help в меню додатку, де вказана інструкція користування даним продуктом. Також, користувач має можливість переглянути розділ about, де написана загальна інформація про програму і про розробника, і при довгому натисканні на елемент кнопку користувач отримає можливість дізнатися функціонал додатку, що допоможе йому краще розбиратися в роботі програми.

Перейшовши в каталог розділу можемо побачити новий добавлений матеріал у вкладці «New». Знімок екрану продемонстровано на рисунку 3.7.

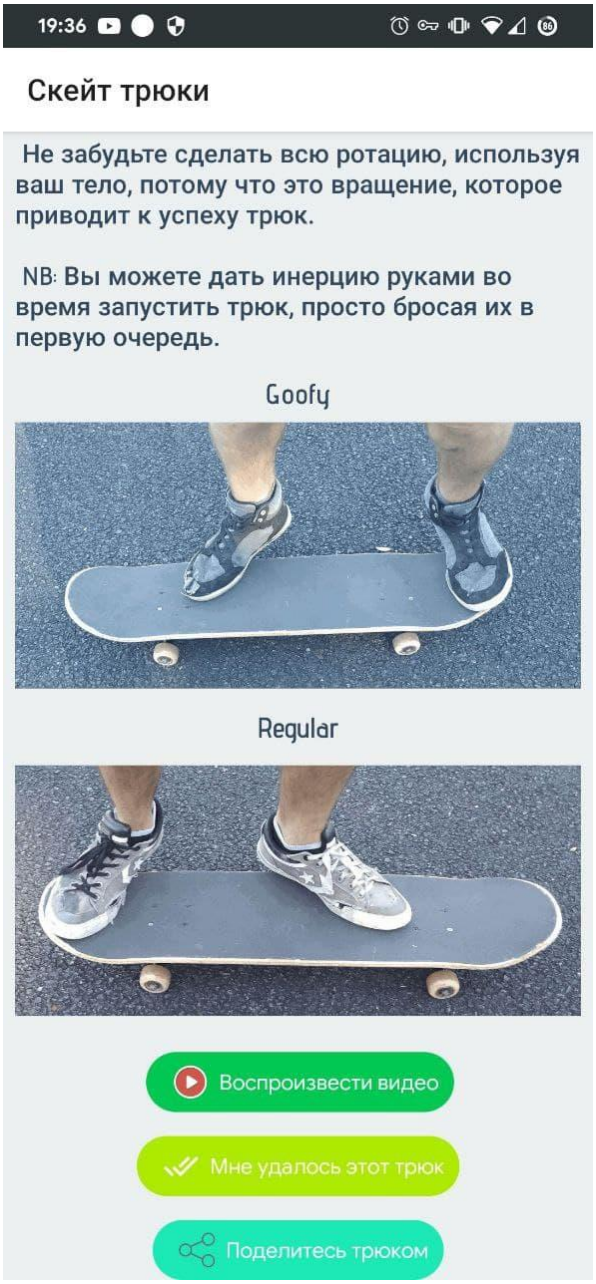


Рисунок 3.7 – Екран каталогу нових розділів

Перейшовши на вкладку «ROLL THE DICE» можемо зіграти в класичну гру СКЕЙТ (рис. 3.8).

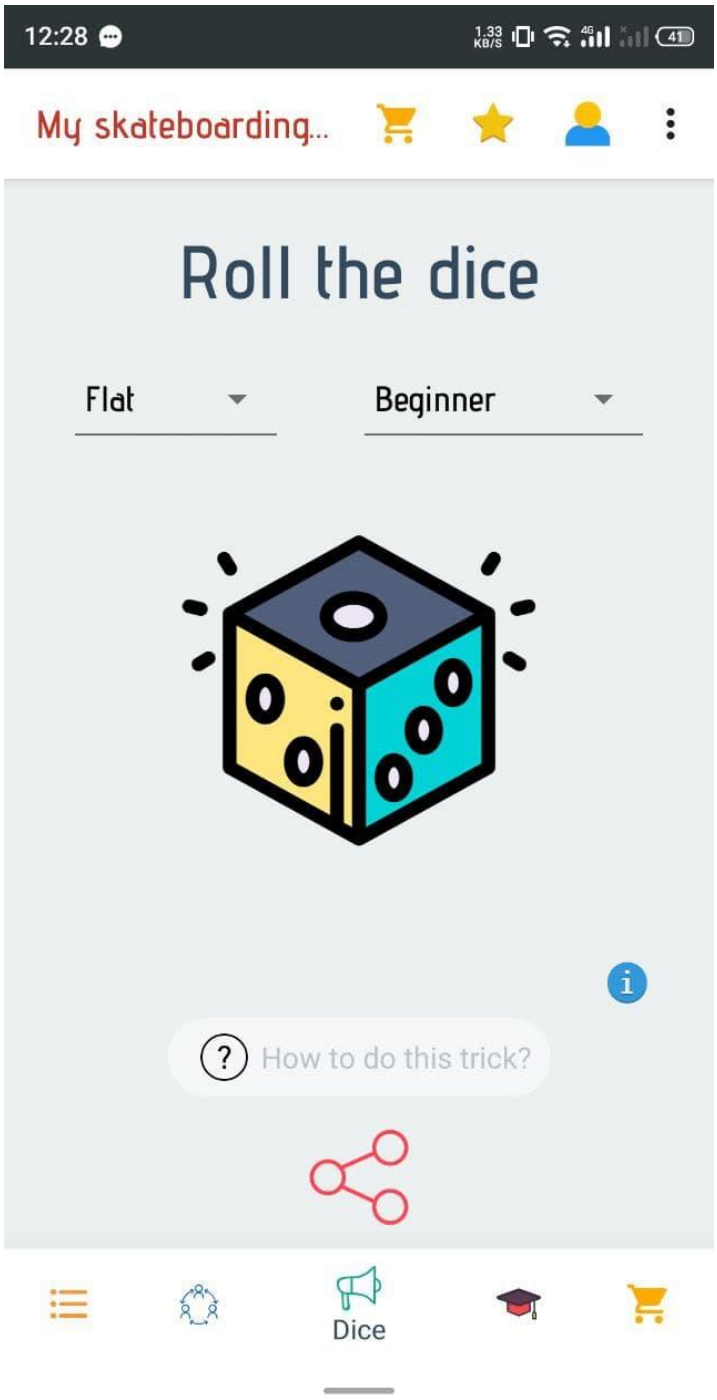


Рисунок 3.8 – Екран каталогу Roll The Dice

Перейшовши на вкладку «Високий рейтинг» отримуємо список трюків (рис. 3.9).

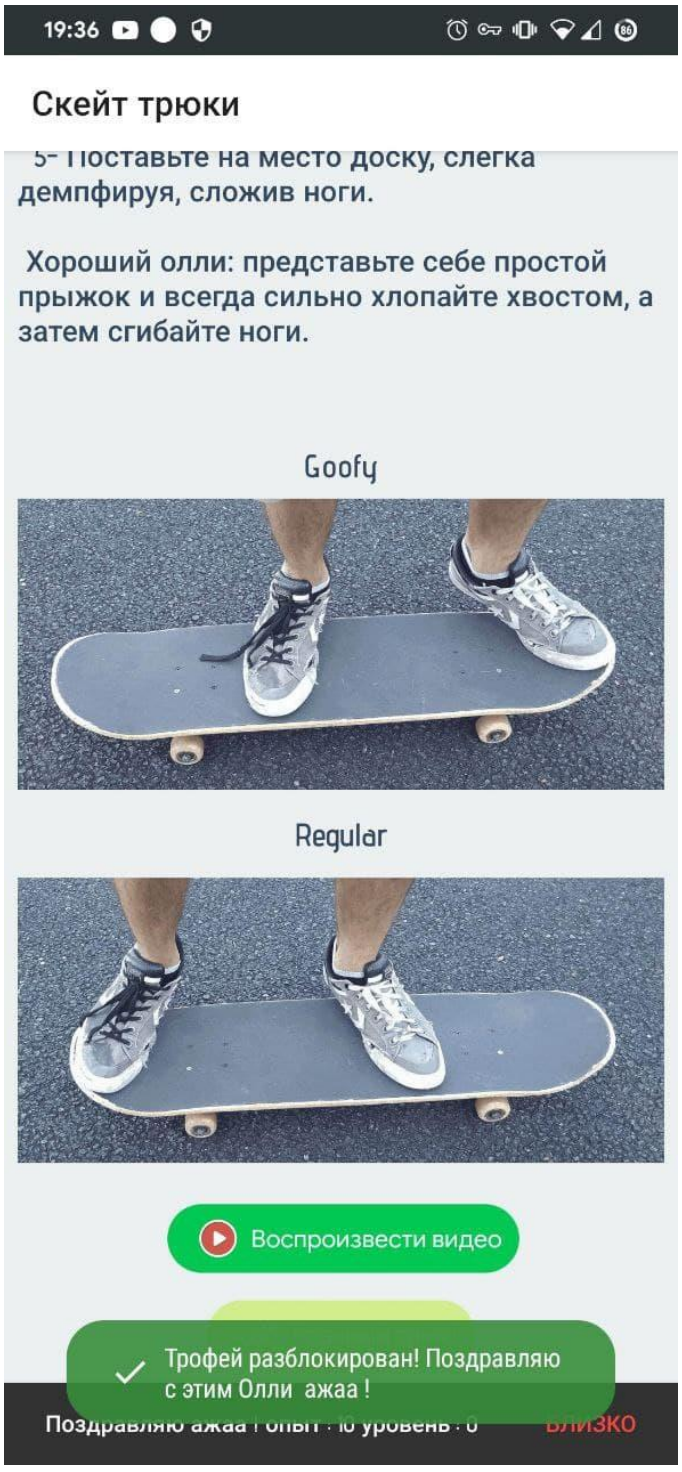


Рисунок 3.9 – Екран каталогу трюків з високим рейтингом

Вкладка «Низький рейтинг» отримуємо список трюків (рис. 3.10).

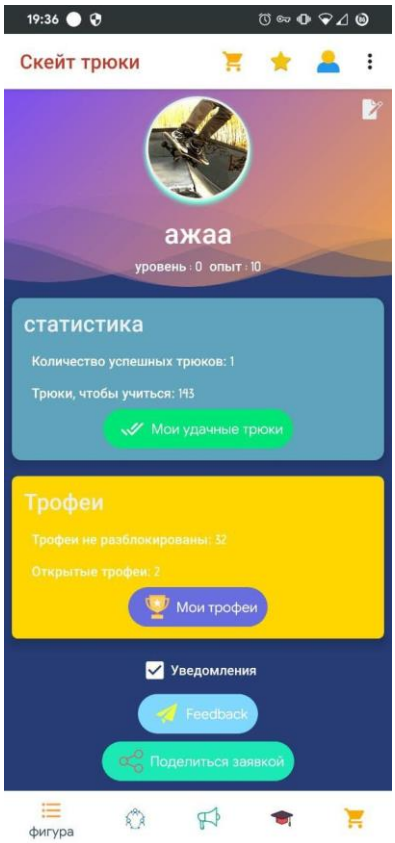


Рисунок 3.10 – Екран каталогу трюків з низьким рейтингом

3.5 Висновки до розділу 3

В даному розділі представлено етапи розробки інформаційної системи та покрокове виконання.

Розробку розпочата із так званого сервера де виділено бізнес-логіку у вигляді сутностей класів, покроково описано розробку пакетів та класів даного модуля.

Наступним етапом розробки був парсер, який відповідно до вимог повинен був розшукувати інформацію та витягувати з веб-сторінок, покрокова реалізація представлена вище у розділі та головним лістингом у додатку Д.

Останнім підпунктом розробка клієнта, який був розроблений за допомогою так званих екранів активності котрі взаємодіють між собою, а також між сервером, отримуючи дані та відображаючи їх.

Дана система протестована за допомогою ручного тестування.

4 ОХОРОНА ПРАЦІ

4.1 Значення охорони праці і навколишнього середовища в забезпечені безпечних і здорових умов праці

Проблема охорони праці набуває особливого значення в умовах сучасного виробничого середовища. Складність технологічних систем та процесів ставить підвищенні вимоги до організму людини, їй доводиться діяти на межі своїх фізичних та психологічних можливостей. В таких умовах людина не завжди може досконало сприймати швидкі зміни обставин в процесі виробничої діяльності і адекватно на них реагувати. Навіть звичайна праця у науковому відділі вже стає небезпечною для здоров'я працівника, тому що при цьому використовуються персональні обчислювальні машини (ПЕОМ), факси, ксерокси та інші прилади, без яких сучасна професійна діяльність неможлива, але всі вони мають високо небезпечні для людини фактори.

На всіх підприємствах, в установах, організаціях створюються безпечні і нешкідливі умови праці.

Небезпечні і шкідливі виробничі фактори виробничого середовища впливають на здоров'я і працездатність людини. Вони можуть бути причиною травм за певних умов.

Причини нещасних випадків поділяються на організаційні, технічні та санітарно-гігієнічні.

Організаційні причини:

– порушення режиму роботи і відпочинку;

– незадовільна організація, розташування і утримання робочих місць, проходів та проїздів;

– використання невідповідного інструмента, обладнання, пристроїв;

– незадовільна якість або відсутність індивідуальних захисних засобів;

– неправильна організація праці, нераціональний режим роботи;

– тривале вимушене одноманітне або ненормальне положення тіла чи окремих його частин та їх перенапруження та інші.

Технічні причини:

– недосконалість технологічних процесів;

– недосконалість обладнання і пристроїв;

– відсутність огорож і запобіжних пристроїв;

– незадовільний стан обладнання, інструмента і пристроїв.

Санітарно-гігієнічні причини:

– недостатність об'єму і площі виробничих приміщень;

– ненормальні метеорологічні умови (температура, вологість, швидкість руху і тиск повітря);

– освітлення не відповідає нормам;

– шкідливі випромінювання.

Аналіз виробничого травматизму проводиться з метою встановлення закономірностей виникнення травм на виробництві та розробки ефективних профілактичних заходів. Аналіз травматизму серед людей, що обслуговують ПЕОМ та працюють на ній, вказує на те, що в основному нещасні випадки трапляються під впливом небезпечних виробничих факторів при недотриманні інструкцій по безпеці праці.

Для аналізу виробничого травматизму застосовують чотири основних методи: статистичний, монографічний, економічний, метод фізичного і математичного моделювання.

За коефіцієнт частоти травматизму $K_ч$ береться кількість нещасних випадків, що припадають на тисячу працівників за певний період:

$$K_ч = \frac{H \cdot 1000}{T}, \qquad 4.1$$

де Н – число нещасних випадків за звітний період (для підприємства Н становить 2);

Т – середньооблікова кількість працівників за той же період (в нашому випадку Т = 45).

$$K_ч = \frac{2 \cdot 1000}{45} = 44.4, \qquad 4.2$$

Коефіцієнт важкості травматизму $K_т$ характеризує середня кількість днів непрацездатності, що припадають на один нещасний випадок:

$$K_т = \frac{D}{H}, \qquad 4.3$$

де D – сумарна кількість днів непрацездатності всіх потерпілих при нещасних випадках за звітний період (55 днів).

$$K_т = \frac{55}{2} = 27,5. \qquad 4.4$$

Зміна коефіцієнтів частоти, важкості і втрат протягом ряду періодів характеризує динаміку промислового травматизму й ефективність заходів щодо попередження травматизму.

Ці коефіцієнти дають можливість вивчати динаміку травматизму на підприємстві, порівнювати його з іншими підприємствами, аналізувати причини травматизму і розробляти заходи для попередження нещасних випадків. Але при цьому не вивчаються ґрунтовно-виробничі умови, при яких сталися нещасні випадки.

До витрат на охорону навколишнього природного середовища належать усі види витрат, спрямовані на запобігання, зменшення чи ліквідацію забруднення, інших видів шкідливого впливу господарської та іншої діяльності на навколишнє природне середовище, при наданні послуг чи використанні продукції.

У трудовому праві прийнято розуміти охорону праці в широкому сенсі як всю сукупність норм законодавства про працю, спрямованих на охорону і захист трудових прав працівників, їх положення в сфері праці.

Основне завдання охорони праці - профілактика та запобігання виробничого травматизму, професійних захворювань і мінімізація соціальних наслідків. Іншими словами, основне завдання охорони праці полягає в тому,

щоб забезпечити па кожному робочому місці соціально прийнятний ризик.

Такі напрями витрат, як економія ресурсів та енергозбереження, ураховуються тільки в тому випадку, коли вони спрямовані передусім на захист охорони навколишнього природного середовища, наприклад утилізацію відходів, яка здійснюється з метою охорони навколишнього природного середовища.

4.2 Аналіз умов праці і виявлення потенційно небезпечних та шкідливих виробничих факторів

Оператори ЕОМ стикаються з впливом таких фізичних і небезпечних психологічних факторів, як підвищена температура, відсутність або недостатність природного світла на робочому місці, електричний струм, статична електрика, розумове перенапруження, перенапруження зорових аналізаторів, монотонність роботи.

Основним джерелом проблем, пов'язаних з охороною здоров'я людей, що використовують у своїй роботі ПК, є дисплеї з електронно-променевими трубками. Вони є джерелами найбільш шкідливих випромінювань, що несприятливо впливають на здоров'я людини. Існує два типи випромінювань, які виникають при роботі монітора: статичне і електромагнітне. Перше виникає при опроміненні екрану потоком заряджених частинок. Неприємності, викликані ним, пов'язані з пилом, що накопичується на електростатичних заряджених екранах, яка летить на людину під час його роботи за дисплеєм. Результати медичних досліджень показали, що така наелектризована пил може викликати запалення шкіри.

При роботі з ПЕОМ можуть виникнути потенційно небезпечні та шкідливі фактори, вплив яких на організм людини може принести йому шкоди і призвести до травматизму.

ПЕОМ встановлюються і розміщуються відповідно до вимог технічних умов заводів-виготовлювачів. Вплив шкідливих електромагнітних

випромінювань зменшується за рахунок видалення їх джерел від оператора і установкою захисного екрана на монітор ПЕОМ. Вплив загазованості, запиленості і шкідливих парів, що виділяються ізоляцією установки усувається за рахунок правильного розміщення обладнання, що забезпечує хорошу природну вентиляцію.

Індекс ізоляції повітряного шуму між залом для глядачів і апаратної звукового забезпечення (при закритих оглядових вікнах) повинен бути не гірше 50 дБ. Стіни апаратної звукового забезпечення та стеля повинні оброблятися звукопоглинальними матеріалами з коефіцієнтом звукопоглинання не менше 0,6 в діапазоні частот 500 - 2000 Гц.

Праця працівників ОЦ повинна відносити до I – II класу за гігієнічними умовами праці: його тяжкість не повинна перевищувати оптимальних, а напруженість допустимих значень. Небезпечні фактори і їх допустимі значення наведені в таблиці 4.1

Таблиця 4.1 - Аналіз потенційних небезпек виробничих факторів

Джерело небезпек	**Характеристика потенційно-небезпечних виробничих факторів та їх допустимі значення**
1 Головний офіс: - електричний струм - електромагнітне поле	Фактичні (середні) дані вимірів: напруга 220-230 В, струм 25 А, частота 50 Гц. Можливість ураження електричним струмом. Діюче значення напруженості ЕМП: Е=30 А/м в діапазоні частот 50 Гц- 100кГц ГДР: Ен=50 В/м в діапазоні частот 60 Гц- 3МГц Н=30 А/м в діапазоні частот 50 Гц- 100кГц ГДР: Нн=5 А/м в діапазоні частот 60 Гц- 3МГц

2 ЕОМ: - ультрафіолетове випромінювання - рентгенівське випромінювання - електростатичне поле - ІЧ – випромінювання - видимий діапазон - яскравість	Фактичні дані вимірів: 0,1Вт/м^2. Допустима інтенсивність: 0,01 Вт/м^2 Фактичні дані вимірів: 15 мкР/год. ГДД: 75мкр.год. Фактичні дані: 15 кВ/м (0 Гц) Допустима напруженість поля 20-60 кВ/м. Фактичні дані вимірів: 0,06-7 Вт/м^2 (в діапазоні 700 нм-1мм). Допустима інтенсивність: 100Вт/м^2 Фактичні дані: 10,5 Вт/м^2 (в діапазоні 4-700 нм). Допустима інтенсивність: 10 Вт/м^2. Фактичні дані: 70 кд/м^2. Допустиме значення: 35 кд/м^2

Для збереження працездатності і попередження розвитку захворювань опорно-рухового апарату користувачів ПЕОМ організовано для них робочі місця, що відповідають вимогам ГОСТ 12.2.032-78.

При тривалій роботі за екраном дисплея в операторів відзначається виражена напруга зорового апарату з появою скарг на незадоволеність роботою, порушення сну, головну біль, дратівливість і хворобливі відчуття в очах, у попереку, руках і ін.

Згідно ГОСТ 12.2.032-78 конструкція робочого місця і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця користувача ПЕОМ дотримано наступні основні умови:

– достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення;

– оптимальне розміщення устаткування;

– природне і штучне освітлення.

4.3 Забезпечення нормальних умов праці

Одним з найважливіших завдань охорони праці є забезпечення таких умов праці, які б вилучали можливість дії на працюючих різного роду небезпечних і шкідливих виробничих факторів.

Згідно зі статтею 153 Кодексу закону про працю, власник підприємства зобов'язаний забезпечити належне технічне об ладнання всіх робочих місць і створювати на них умови праці відповідно до нормативних актів з охорони праці.

Приміщення являє собою кімнату площею 21м^2 та об’ємом 56,7м^3. Кількість робочих місць у кімнаті – два. На одного працівника припадає 28,35м^3 об’єму приміщення та 10,5м^2 площі, що задовольняє вимогам “Державних санітарних правил та норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98”, п.2 “Вимоги до виробничих приміщень для експлуатації ВДТ ЕОМ та ПЕОМ”, згідно з якими площа на одне робоче місце має становити не менше ніж 6 м^2, а об'єм не менше ніж 20м^3.

Роботи відносяться до легкого характеру роботи категорії Іа, що виконуються сидячи й не потребують фізичного напруження.

При нормуванні умов для різних галузей промисловості виходять із загальних міжгалузевих норм

ГОСТ 12.1.005-88 “Загальні санітарно-гігієнічні вимоги до повітря робочої зони”. На сьогодні основним нормативним документом, що визначає параметри мікроклімату виробничих приміщень є санітарні норми ДСН 3.3.6.042-99.

Оптимальні параметри мікроклімату у робочій зоні виробничого

приміщення в теплий та холодний періоди року наведені в таблиці 4.2.

Таблиця 4.2 – Оптимальні параметри мікроклімату

Назва приміщення	**Категорія важкості фізичних робіт**	**Період Року**	**Температура ^{o}C**	**Відносна вологість %**	**Швидкість руху**
Приміщення з ЕОМ	Легка-1а	Холодний	22-24	40-60	0.1
	Легка-1а	Теплий	23-25	40-60	0.1

Кожна будівля чи окреме приміщення характеризуються, в залежності від властивостей їх зовнішніх захищень певною величиною максимальних тепловтрат. При розрахунку тепловтрат будівлі беруться до уваги мінімальні температурні показники для даної місцевості.

У приміщенні обладнана система опалення та кондиціювання повітря відповідно до СНиП 2.04.05-91. Використовується кондиціонер EWT G-091GS. Характеристики наведені в таблиці 4.3

Таблиця 4.3 – Характеристики кондиціонера EWT G-091GS

Характеристики по холоду	
Площа що обслуговується (м2) Споживана потужність, кВт Коефіцієнт ефективності, EER Гарантований діапазон зовнішніх температур	25 1,01 2,61 +21 - +43
Характеристики по теплу	
Площа що обслуговується (м2) Споживана потужність, кВт Коефіцієнт ефективності, COP Гарантований діапазон зовнішніх температур	27 1,07 2,61 -5 - +24
Технічні характеристики внутрішнього блоку	
Рівень шуму (мін-макс), дБ (А) Вага, кг Габарити (ВхШхГ), мм	27-35 8 255x730x174
Технічні характеристики зовнішнього блоку	
Рівень шуму (мін-макс), дБ (А) Вага, кг Габарити (ВхШхГ), мм	50 33 430x720x310
Загальні характеристики	

Джерело живлення, В / фаза / Гц	220-240/1/50
Максимальна довжина магістралі, м	10
Максимальний перепад висот, м	5
Фреон	R22

Для підтримання в приміщеннях нормальних параметрів повітряного середовища, яке відповідає санітарно-гігієнічним і технологічним вимогам, влаштовують вентиляцію.

Таблиця 4.4 – Характеристика системи вентиляції

Приміщення	**Тип вентиляції**	**Вентиляційне обладнання**	**Кратність повітряного обміну**
Приміщення для експлуатації ЕОМ	Комбінована	Кондиціонер EWT G-091GS	3

Важливе місце в комплексі заходів з охорони праці працюючих з ПК займає утворення оптимального світлого середовища, тобто, раціональна організація природного і штучного освітлення приміщення і робочих місць.

При незадовільному освітленні знижується продуктивність праці користувачів ЕОМ, можлива поява короткозорості, швидка стомлюваність.

Система освітлення відповідає таким вимогам:

– освітленість на робочому місці відповідає характеру зорової роботи, який визначається трьома параметрами: об'єктом розрізнення - найменшим розміром об'єкта, що розглядається на моніторі персонального комп'ютера (ПК) та робочої станції; фоном, який характеризується коефіцієнтом відбиття; контрастом об'єкта і фону;

– забезпечено достатньо рівномірний розподіл яскравості на робочій поверхні монітора, а також в межах навколишнього простору;

– на робочій поверхні відсутні різкі тіні;

– у полі зору відсутні відблиски (підвищеної яскравості поверхонь, які світяться та викликають осліплення);

– величина освітленості є постійною під час роботи.

Джерела світла відносно робочого місця розташовують таким чином, щоб включити влучення в очі прямого світла. Захисний кут арматури в цих джерелах становить більше 30^0.

Розраховується площа світлопрорізів наступним чином:

- при боковому освітленні приміщень за формулою наведеною нижче:

$$S_{\text{в}} = \frac{D_{\text{н}}}{100\,m} \cdot \frac{K_{\text{з}}\eta_{\text{в}}K_{\text{буд}}}{\tau_o r_1} \qquad 4.5$$

- при верхньому освітленні приміщень за формулою наведеною нижче:

$$S_{\text{л}} = \frac{D_{\text{н}}}{100\,m} \cdot \frac{K_{\text{з}}\,\eta_{\text{л}}}{\tau_o r_1 K_{\text{л}}} \cdot S_n \qquad 4.6$$

Де:

$S_{\text{в}}$ і $S_{\text{л}}$ – площі світлових прорізів (в світлі) відповідно при боковому та верхньому освітленні;

S_n – площа підлоги приміщення, яка = 21 м^2;

$D_{\text{н}}$ – нормоване значення КПО, $D_{\text{н}} = 3$;

m – коефіцієнт світлового клімату світлопрорізу, який згідно даного регіону м. Івано-Франківськ m = 1,02

$К_{\text{з}}$- коефіцієнт запасу. згідно з ДБН 3.25-28-2018 для даної категорії приміщень $К_{\text{з}}$ = 1,2;

$\eta_{\text{л}}\,\eta_{\text{в}}$ – коефіцієнти, що враховують світлову активність вікон та ліхтарів, які $\eta_{\text{л}}$= 7,8 $\eta_{\text{в}}$ =8,5;

$K_{\text{л}}$ – коефіцієнти, що враховує тип ліхтарів, $K_{\text{л}}$=1,2;

$K_{\text{буд}}$ – коефіцієнт, що враховує затемнення вікон будівлями, що знаходяться напроти, $K_{\text{буд}}$ = 1,0;

$r_1\,r_2$ – коєфіцієнти, що враховується підвищення КПО за рахунок світла, відбитого від внутрішніх поверхонь приміщення, які r_1 =1,6 , $r_2 = 1{,}2$;

τ^0 - загальний коефіцієнт світлопропускання, який розраховується за формулою:

$$\tau_0 = \tau_1 \cdot \tau_2 \cdot \tau_3 \cdot \tau_4 \cdot \tau_5, \qquad 4.7$$

τ_1 - коефіцієнт світлопропускання матеріалу вікон, τ_1=0,9;

τ_2 - коефіцієнт, що враховує витрату світла, для металопластикових τ_2 = 0,75;

τ^3 - коефіцієнт, що враховує витрату світла в несучих конструкціях, τ^3 = 1,2;

τ_4 - коефіцієнт, що враховує витрату світла в сонцезахисних пристроях, τ_4 = 1,0;

τ^5 - коефіцієнт, що враховує витрату світла в захисній сітці, τ^5 = 1.

Тоді:

- коефіцієнт світлопропускання розрахований:

$$\tau_o = 0{,}9 \cdot 0{,}75 \cdot 1{,}2 \cdot 1{,}0 \cdot 1 = 0{,}81$$

- розраховуємо при боковому освітленні приміщень:

$$S_в = \frac{3}{100 \cdot 1{,}02} \cdot \frac{1{,}2 \cdot 8{,}5 \cdot 1}{0{,}81 \cdot 1{,}6} = 0{,}03 \cdot 7{,}87 = 2{,}36$$

- розраховуємо при верхньому освітленні:

$$S_л = \frac{3}{100 \cdot 1{,}02} \cdot \frac{1{,}2 \cdot 7{,}8}{0{,}81 \cdot 1{,}6 \cdot 1{,}2} \cdot 21 = 0{,}03 \cdot 6 \cdot 21 = 3{,}78$$

Відомо, що вісімдесят відсотків інформації зовнішнього світу людина отримує через очі. Якість інформації залежить від освітлення. Тому правильно організована система освітлення має велике значення в зниженні виробничого травматизму, створює нормальні умови для роботи органів зору, підвищує працездатність організму і відповідно, продуктивність праці: при зорових роботах середньої важкості на 5…6%, при важкій зоровій роботі на 15%, а при роботі в межах зорового сприйняття – на 40%.

Відповідно площі світлових прорізів (в світлі)при боковому освітленні $S_в = 2{,}36$ та верхньому освітленні $S_л = 3{,}78$

Скористаємося методом використання світлового потоку. Для визначення потрібної кількості світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F = \frac{E \cdot K \cdot S \cdot Z}{\eta}, \qquad 4.8$$

де F – світловий потік, що розраховується, лм;

E – нормована мінімальна освітленість, Лк; E = 300 лк;

S – площа освітлюваного приміщення (у нашому випадку S=21м^2);

Z – відношення середньої освітленості до мінімальної (Z =1,1);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації;

η – коефіцієнт використання світлового потоку освітлювальної системи, який залежить від розподілу сили світла світильників, показника приміщення та коефіцієнтів відбиття стелі, стін і робочої поверхні.

Показник приміщення визначається за формулою:

$$i = \frac{a \cdot b}{h(a+b)}, \qquad 4.9$$

де а, в - ширина і довжина приміщення відповідно, а = 3,5м, в = 6м;

h - висота розташування світильників, h = 2,7м. Підставивши значення отримаємо:

$$i = \frac{3{,}5 \cdot 6}{2{,}7(3{,}5+6)} \approx 0{,}8$$

Для світильників типу ОДО при i = 0,8, $\rho_{стелі}$ = 50%, $\rho_{стін}$ = 30%, $\rho^{роб}$ =30%, ŋ = 0,36. Світловий потік буде дорівнювати:

$$\Phi = \frac{300 \cdot 21 \cdot 1{,}5 \cdot 1{,}1}{0{,}36} = 28875 лм.$$

Визначаємо кількість світильників за формулою:

$$n = \frac{\Phi}{\Phi_л}, \qquad 4.10$$

де $\Phi_л$ - світловий потік однієї лампи.

Для ламп Philips TLD світловий потік становить $Ф_л$ = 2800 лм.

$$n = \frac{28875}{2800} \approx 10.$$

В приміщенні кожен світильник комплектується двома лампами. Тобто необхідно використовувати 5 світильників із 10 працюючими лампами в них. Загальну робочу освітленість визначаємо за формулою:

$$E = \frac{Ф_л \cdot n \cdot \eta}{S \cdot K \cdot Z} = \frac{2800 \cdot 10 \cdot 0{,}36}{21 \cdot 1{,}5 \cdot 1{,}1} = 300 лк.$$

Дане розрахункове значення знаходиться в межах 300-400лк, встановлених ДБН 3.25-28-2018.

4.4 Забезпечення безпеки технологічних процесів, монтажу, пусконалагоджувальних, ремонтних робіт та експлуатації обладнання, приладів та пристроїв.

При проектуванні систем електропостачання, при монтажі силового електроустаткування і електричного освітлення в будівлях і приміщеннях для ЕОМ необхідно дотримуватися вимог нормативно-технічної документації (ПУЕ, ПТЕ, ПТБ і ін.).

ЕОМ є однофазним споживачем електроенергії, що живиться від трифазної чотирьох проводної мережі з глухо заземленою нейтраллю змінного струму частотою 50 Гц. Електробезпека ЕОМ забезпечується комплексом конструктивних, схемноконструктивних і експлуатаційних засобів і способів захисту [46].

Експлуатаційні заходи захисту ґрунтуються на дотриманні правил техніки безпеки при роботі з високою напругою і наступних запобіжних засобів:

– не підключати і не відключати роз'єми кабелів при включеній напрузі мережі;

– технічне обслуговування і ремонтні роботи проводити тільки при вимкненому живленні мережі.

Конструктивні заходи безпеки забезпечують захист від випадкового дотику до струмопровідних частин за допомогою захисних оболонок і ізоляції струмопровідних елементів. Ступінь захисту оболонки ЕОМ повинен відповідати класу пожежонебезпечної зони приміщення П-IIа [47] і бути не нижчим IP-44. В мережах напругою до 1000В з глухо заземленною нейтрально як схемно-конструктивною мірою захисту застосовується занулення.

Експлуатаційні заходи.

Необхідно дотримувати правила техніки безпеки при роботі з високою напругою і наступні запобіжні засоби:

– монтаж, обслуговування, ремонт і наладка ЕОМ, заміна деталей, пристосувань і блоків повинна здійснюватися тільки при повному виключенні живлення;

– в приміщеннях, де експлуатуються більше 5 комп'ютерів на видному і доступному місці встановлюється аварійний і резервний вимикач для повного відключення електроживлення;

– заземлені конструкції в приміщенні повинні бути надійно захищений діелектричними щитками або сітками від випадкового дотику.

Працівник, що поступає на роботу, обов'язково проходить вступний і первинний інструктаж по техніці безпеки в цілях профілактики нещасних випадків, а також знайомиться з інструктажем по дотриманню заходів техніки безпеки при роботі з ПЕВМ.

При організації праці, пов'язаної з використанням ПК, для збереження здоров'я працюючих, запобігання професійним захворюванням І підтримки працездатності передбачаються внутрішньо змінні регламентовані перерви для відпочинку.

Внутрішньозмінні режими праці й відпочинку містять додаткові нетривалі перерви в періоди, що передують появі об'єктивних і суб'єктивних ознак стомлення й зниження працездатності.

При виконанні робіт, що належать до різних видів трудової діяльності, за основну роботу з ПК слід вважати таку, що займає не менше 50% робочого часу. Впродовж робочої зміни мають передбачатися:

– перерви для відпочинку і вживання їжі (обідні перерви);

– перерви для відпочинку й особистих потреб (згідно із трудовими нормами);

– додаткові перерви, що вводяться для окремих професій з урахуванням особливостей трудової діяльності.

За характером трудової діяльності розрізняють три професійні групи, згідно з діючим класифікатором професій:

– розробники програм інженери-програмісти виконують роботу переважно з відеотерміналом та документацією при необхідності інтенсивного обміну Інформацією з ЕОМ і високою частотою прийняття рішень. Робота характеризується інтенсивною розумовою творчою працею з підвищеним напруженням зору, концентрацією уваги на фоні нервово-емоційного напруження, вимушеною робочою позою, загальною гіподинамією, періодичним навантаженням на кисті верхніх кінцівок. Робота виконується в режимі діалогу з ПК у вільному темпі з періодичним пошуком помилок в умовах дефіциту часу;

– оператори електронно-обчислювальних машин виконують роботу, пов'язану з обліком інформації, одержаної із ВДТ за попереднім запитом, або тієї, що надходить з нього, супроводжується перервами різної тривалості, пов'язана з виконанням іншої роботи й характеризується напруженням зору, невеликими фізичними зусиллями, нервовим напруженням середнього ступеня та виконується у вільному темпі;

– оператор комп'ютерного набору виконує одноманітні за характером роботи з документацією та клавіатурою і нечастими нетривалими переключеннями погляду на екран дисплея, з введенням даних з високою швидкістю. Робота характеризується як фізична праця з підвищеним навантаженням на кисті верхніх кінцівок на фоні загальної гіподинамії з

напруженням зору (фіксація зору переважно на документи), нервово-емоційним напруженням.

Правилами встановлюються такі внутрішньозмінні режими праці та відпочинку при роботі з ПК при 8-годинній денній робочій зміні в залежності від характеру праці:

– для розробників програм із застосуванням ПК слід призначати регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за ПК;

– для операторів із застосуванням ПК слід призначати регламентовані перерви для відпочинку тривалістю 15 хвилин через кожні дві години;

– для операторів комп'ютерного набору слід призначати регламентовані перерви для відпочинку тривалістю 10 хвилин після кожної години роботи за ПК.

У всіх випадках, коли виробничі обставини не дозволяють застосувати регламентовані перерви, тривалість безперервної роботи з ПК не повинна перевищувати 4 години.

При 12-годинній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин робота аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

Для зниження нервово-емоційного напруження, втомлення зорового аналізатора, поліпшення мозкового кровообігу, подолання несприятливих наслідків гіподинамії, запобігання втомі доцільно деякі перерви використовувати для виконання комплексу вправ, які наведені у Державних санітарних правилах і нормах роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСаяПІН 3.3.2.018.

ВИСНОВКИ

В дипломному проекті було розроблено «Мобільний додаток для трекінгу особистого прогресу в скейтбордингу» для навчання.

Мобільний додаток для трекінгу особистого прогресу в скейтбордингу.

Додаток має дві основні вкладки - профіль та навчання.

У Вкладці «профіль» відображається особистий прогрес, тобто особистий рівень (початківець, любитель, продвинутий, про), кількість трюків які вивчені, та має бути можливість переглянути які трюки вже вивчені. Окрім цього відображається фото користувача,його нікнейм та стаж катання.

У Вкладці «навчання» мають бути 5 розділів (Основи, шовіти, фліпи, слайди та гранди, греби). Всі трюки розподіляються по певним розділам і вивчаються від легшого до складнішого, але не обов'язково, можно переходити вручну. Кожна сторінка з вивчення трюку має текстове поле, гіф з відображенням трюку та відео урок підтягнутий з ютубу

При розробці обрано актуальні технології, розроблено серверну частину яка відповідає за обробку та зберігання даних.

Парсер, який відслідковує дані актуальних пропозицій, формує список та посилає серверу.

Клієнт у виді додатку з інтуїтивним інтерфейсом, який є наочним, зрозумілим і доступним для користувачів. Створено меню для навігації по додатку та переходу між екранами:

Створено форму входу та реєстрації в додаток.

Для створення інформаційної системи використовувались актуальні технології.

Слід зазначити що даний дипломний проект є актуальним та може розвиватися в майбутньому.

Дана система направлена на полегшення навчання, складання списку трюків та загалом для полегшення життя людям, а саме для тих хто хоче оволодіти скейтбордом.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Android 2. Програмування додатків для планшетних комп'ютерів і смартфонів (Рето Маєр. - СПб .: Санкт-Петербург, 2011. - 672 с.)
2. Android. Програмування додатків під операційну систему Google (Лорен Дерсі, Рид Групп, 2011, 831 с.).
3. Android 2. Програмування додатків для планшетних комп'ютерів і смартфонів (Рето Маєр, Ексмо, 2011, 356 с.).
4. Android. Програмування для професіоналів 2-е видання (Ед Бурнет, 2016р. , 755с.).
5. Google Android створення додатків для смартфонів і планшетних ПК (А. Голощапов. 2013, с. 341).
6. Книга Привіт, android! розробка мобільних додатків.(Ед Бурнет СПб: 2012 Пітер, 256 с.)
7. Розробка додатків для Android (С. Хашимі, С. Коматінені, Д. Маклін СПб.: Питер, 2011. — 736 с.).
8. Форум про програмування для мобільних пристроїв [Електронний ресурс]], Режим доступу: http://www.4pda.ru
9. Офійний сайт AndroidStudio – [Електронний ресурс], Режим доступу: https://developer.android.com/studio/preview/index.html
10. Статті про програмування для Android [Електронний ресурс] Режим доступу: http://flashbot.ru/android-dev
11. Сторінка підтримуваних технологій – [Електронний ресурс], Режим доступу: https://developer.android.com/studio/features.html
12. Сторінка створення проекту на ОС Android – [Електронний ресурс], Режим доступу: https://developer.android.com/studio/index.html
13. Сторінка створення послідовності пакетів – [Електронний ресурс], Режим доступу: https://developer.android.com/studio/intro/index.html
14. Створення макетів для інтерфейсу – [Електронний ресурс], Режим доступу: https://developer.android.com/design/material/index.html?hl=ru

15. Калічак О. В. Електросилова та електроосвітлювальне устаткування. (Київ,2015р. , 231 с.)

16. Програмування для Android. Самовчитель (Колісниченко Д. - СПб .: Санкт-Петербург, 2011. - 736 с.)

17. Правила улаштування електроустановок. Розділ 6. Електричне освітлення. (Київ,2016р. , 521 с.)

18. ДБН В.2.5-28 Природне і штучне освітлення. Київ, 2018.

19. 4. Справочная книга для проектирования электрического освещения / Под. редакцией Г.М. Кноринга / - Л.: Энергия, 2016.

20. Козинский В.А. Электрическое освещение и облучение: -М.: Агропромиздат, 2011.

ДОДАТКИ

Додаток А

Код класу Account:

```
@Entity
@Table(name = "account")
public class Account {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;
    @Column(name = "username")
    private String username;
    @Column(name = "password")
    private String password;
    @ManyToMany(cascade = CascadeType.ALL)
    @JoinTable(name = "shoplist", joinColumns = @JoinColumn(name
= "account_id", referencedColumnName = "id"),
            inverseJoinColumns = @JoinColumn(name = "item_id",
referencedColumnName = "id"))
    private Set<Item> items;
    public Set<Item> getItems() {
        return items;
    }
    public void setItems(Set<Item> items) {
        this.items = items;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Код класу Item:

```
@Entity
@Table(name = "item")
public class Item {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private int id;
    @Column(name = "name")
    private String name;
    @Column(name = "category")
    private String category;
    @Column(name = "image_url")
    private String imageUrl;
    @Column(name = "old_price")
    private Double oldPrice;
    @Column(name = "new_price")
    private Double newPrice;
    @Column(name = "discount")
    private String discount;
    @Column(name = "date_in")
    private Date dateIn;
    @Column(name = "date_out")
    private Date dateOut;
    @Column(name = "crawl_date")
    private Date crawlDate;
    @Column(name = "condition")
    private String condition;
    //   private Image image;
    @JsonIgnore
    @ManyToMany(mappedBy = "items")
    private Set<Account> accounts;
    public Set<Account> getAccounts() {
        return accounts;
    }
    public void setAccounts(Set<Account> accounts) {
        this.accounts = accounts;
    }
    public Item() {
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```
        this.name = name;    }
    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public String getImageUrl() {
        return imageUrl;
    }
    public void setImageUrl(String imageUrl) {
        this.imageUrl = imageUrl;
    }
    public Double getOldPrice() {
        return oldPrice;
    }
    public void setOldPrice(Double oldPrice) {
        this.oldPrice = oldPrice;
    }
    public Double getNewPrice() {
        return newPrice;
    }
    public void setNewPrice(Double newPrice) {
        this.newPrice = newPrice;
    }
    public String getDiscount() {
        return discount;
    }
    public void setDiscount(String discount) {
        this.discount = discount;
    }
    public Date getDateIn() {
        return dateIn;
    }
    public void setDateIn(Date dateIn) {
        this.dateIn = dateIn;
    }
    public Date getDateOut() {
        return dateOut;
    }
    public void setDateOut(Date dateOut) {
        this.dateOut = dateOut;
    }

    public Date getCrawlDate() {
        return crawlDate;
    }

    public void setCrawlDate(Date crawlDate) {
        this.crawlDate = crawlDate;
}
```

```
    public String getCondition() {
        return condition;
    }

    public void setCondition(String condition) {
        this.condition = condition;
    }

    @Override
    public String toString() {
        return "Item{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", category='" + category + '\'' +
                ", imageUrl='" + imageUrl + '\'' +
                ", oldPrice=" + oldPrice +
                ", newPrice=" + newPrice +
                ", discount=" + discount +
                ", dateIn=" + dateIn +
                ", dateOut=" + dateOut +
                ", crawlDate=" + crawlDate +
                ", condition='" + condition + '\'' +
                '}';
    }
}
```

Додаток Б

Код класу AccountService:

```
@Service
public class AccountService implements UserDetailsService {
    @Autowired
    AccountRepository accountRepository;
    @Autowired
    ItemRepository itemRepository;
    public Set<Item> getShoplist() {
        return getCurrentAccount().getItems();
    }
    public void addItemToShopList(int id) {
        Account account = getCurrentAccount();
        Item item = itemRepository.getItemById(id);
        if (item == null) {
            return;        }
        account.getItems().add(item);
        accountRepository.save(account);}
    public void clearShopList() {
        Account account = getCurrentAccount();
        account.getItems().clear();
        accountRepository.save(account);    }
    public void deleteItemFromShopList(int id) {
        Account account = getCurrentAccount();
        account.getItems().removeIf(item -> item.getId() == id);
        accountRepository.save(account);}
    private Account getCurrentAccount() {
        Authentication auth =
SecurityContextHolder.getContext().getAuthentication();
        User user = (User) auth.getPrincipal();
        return
accountRepository.findByUsername(user.getUsername());}
    @Override
    public UserDetails loadUserByUsername(String username)
throws UsernameNotFoundException {
        Account account =
accountRepository.findByUsername(username);
        if (account != null) {
            return new User(account.getUsername(),
                    account.getPassword(),
                    true,
                    true,
                    true,
                    true,
                    AuthorityUtils.createAuthorityList("USER"));
        } else {
            throw new UsernameNotFoundException("Could not find
the user '"
                    + username + "'");
        }
    }
```

```
}
```

Код класу ItemService:

```
@Service
public class ItemService {

    @Autowired
    private ItemRepository itemRepository;

    public List<Item> getCurrentItems() {
        Date dateNow = new Date(System.currentTimeMillis());
        return
itemRepository.findByDateInLessThanEqualAndDateOutGreaterThanEqu
al(dateNow, dateNow);
    }

    public Set<String> getCurrentCategories() {
        Set<String> set = new HashSet<>();
        List<Item> items = getCurrentItems();
        for (Item item : items) {
            set.add(item.getCategory());
        }
        return set;
    }

    public Item addItem(Item item) {
        ExampleMatcher matcher = ExampleMatcher.matching()
                .withIgnorePaths("id", "crawlDate");
        Item found = itemRepository.findOne(Example.of(item,
matcher));
        if (found != null) {
            return found;
        }
        return itemRepository.save(item);
    }
    public Object getInfo() {
        return new Object() {
            public static final int ITEMS_PER_PAGE = 30;
            public int getItemCount() {
                return
ItemService.this.getCurrentItems().size();
            }
            public int getItemsPerPage() {
                return ITEMS_PER_PAGE;
            }
            public int getNumPages() {
                return (int) Math.ceil(getItemCount() / (double)
ITEMS_PER_PAGE);
            }
        };
    }}
```

Додаток В

Код класу MainActivity:

```
public class MainActivity extends AppCompatActivity {
    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;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayShowTitleEnabled(false);
        swipeRefreshLayout = findViewById(R.id.swipeContainer);

swipeRefreshLayout.setOnRefreshListener(refreshListener);
        btnAdd = findViewById(R.id.action_add);
        btnAdd.setOnClickListener(addShopList);
        btnLoginLogout = findViewById(R.id.action_login_logout);
        btnLoginLogout.setOnClickListener(loginLogoutListener);
        fetchData = new FetchData(this, swipeRefreshLayout);
        categories = new ArrayList<>();
        FrameLayout content =
findViewById(R.id.fragmentContainer);
        fragmentManager = getSupportFragmentManager();
        Fragment fragment =
fragmentManager.findFragmentByTag(TAG_FRAGMENT_ONE);
        if (fragment == null) {
            fragment = HomeFragment.newInstance();        }
        replaceFragment(fragment, TAG_FRAGMENT_ONE);
        currentFragmentId = fragment.getId();
```

```
        BottomNavigationView bottomNav =
findViewById(R.id.bottomNavigation);

bottomNav.setOnNavigationItemSelectedListener(navListener);
bottomNav.setOnNavigationItemReselectedListener(reselectNavListe
ner);HomeFragment())
        adapter = new ItemAdapter(new ArrayList<Item>(), this);
        shopListsPreviewAdapter = new
ShopListsPreviewAdapter(new ArrayList<ShopList>(), this);
        shops = new ArrayList<>();
        queue = Volley.newRequestQueue(this);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            ((ProgressBar)
findViewById(R.id.pbLoading)).setProgress(0, true);
        } else {
            ((ProgressBar)
findViewById(R.id.pbLoading)).setProgress(0);        }
        if (!isLoggedIn()) {
            final AlertDialog.Builder alertDialog = new
AlertDialog.Builder(MainActivity.this);
            alertDialog.setTitle(R.string.title_hello_there);
            alertDialog.setMessage(R.string.hello_there);
            alertDialog.setPositiveButton(R.string.okk,
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {}    });
            alertDialog.show();        }
        if
(InternetUtil.isConnectedToInternet(getApplicationContext()))
{etchData.execute(true);
        } else {
            Toast.makeText(this, R.string.noInternetToast,
Toast.LENGTH_LONG).show();
            startActivityForResult(new
Intent(Settings.ACTION_SETTINGS), 0);        }    }
    public boolean isLoggedIn() {
        return !SharedPrefsUtil.getStringPref(this,
Config.KEY_TOKEN).equals(Config.DEF_NO_TOKEN);
    }
    private
BottomNavigationView.OnNavigationItemReselectedListener
reselectNavListener = new
BottomNavigationView.OnNavigationItemReselectedListener() {
        @Override
        public void onNavigationItemReselected(@NonNull MenuItem
item) {
            if (item.getItemId() == R.id.nav_home) {
                int pos = ((LinearLayoutManager)
(((RecyclerView) findViewById(R.id.itemList))

.getLayoutManager())).findFirstVisibleItemPosition();
                int offset = 10;
                if (pos > offset) {
```

```
                    ((RecyclerView)
findViewById(R.id.itemList)).smoothScrollToPosition(pos -
offset);}
                Handler mHandler = new Handler();
                Runnable codeToRun = new Runnable() {
                    @Override
                    public void run() {
                        ((RecyclerView)
findViewById(R.id.itemList)).scrollToPosition(0);
}
                };
                mHandler.postDelayed(codeToRun, 200);}   }    };
    private SwipeRefreshLayout.OnRefreshListener refreshListener
= new SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            if (!homeActive) {
                fetchData.execute(false);
                fetchData.afterDownload();
            } else {
                adapter.addAll(new ArrayList<Item>()); // Clear
item list if refreshed.
                currentPage = 0;
                fetchData.execute(true);
                fetchData.afterDownload();  }        }    };
    @Override
    protected void onResume() {
        super.onResume();
        btnAdd = findViewById(R.id.action_add);
        if (isLoggedIn()) {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

btnLoginLogout.setBackground(getDrawable(R.drawable.ic_logout_bl
ack_24dp));            }        } else {
            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

btnLoginLogout.setBackground(getDrawable(R.drawable.ic_login_bla
ck_24dp));            }        }
        if (homeActive) {
            btnAdd.setVisibility(View.INVISIBLE);
        } else {
            btnAdd.setVisibility(View.VISIBLE);        }
    }
    private
BottomNavigationView.OnNavigationItemSelectedListener
navListener = new
BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull
MenuItem item) {
            btnAdd = findViewById(R.id.action_add);
```

```
            btnLoginLogout =
findViewById(R.id.action_login_logout);
            switch (item.getItemId()) {
                case R.id.nav_home: {
                    Fragment fragment =
fragmentManager.findFragmentByTag(TAG_FRAGMENT_ONE);
                    if (fragment == null) {
                        fragment = HomeFragment.newInstance();
                    }
                    replaceFragment(fragment, TAG_FRAGMENT_ONE);
                    if (btnAdd != null) {
                        btnAdd.setVisibility(View.INVISIBLE);
                    }
                    if (btnLoginLogout != null) {

btnLoginLogout.setVisibility(View.VISIBLE);
                        if (isLoggedIn()) {
                            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

btnLoginLogout.setBackground(getDrawable(R.drawable.ic_logout_bl
ack_24dp));
                            }
                        } else {
                            if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.LOLLIPOP) {

btnLoginLogout.setBackground(getDrawable(R.drawable.ic_login_bla
ck_24dp));
                            }
                        }
                    }
                    break;
                }
                case R.id.nav_shoplist: {
                    Fragment fragment =
fragmentManager.findFragmentByTag(TAG_FRAGMENT_TWO);
                    if (fragment == null) {
                        fragment =
ShopListsPreviewFragment.newInstance();
                    }
                    replaceFragment(fragment, TAG_FRAGMENT_TWO);
                    if (btnAdd != null) {
                        btnAdd.setVisibility(View.VISIBLE);
                    }
                    break;
                }
            }

            if (!MainActivity.this.isLoggedIn()) {
                btnAdd.setVisibility(GONE);
            }
            return true;
```

```
        }
    };

    private void replaceFragment(@NonNull Fragment fragment,
@NonNull String tag) {
        if (!fragment.equals(currentFragmentId)) {
            fragmentManager
                    .beginTransaction()
                    .replace(R.id.fragmentContainer, fragment,
tag)
                    .commit();
            currentFragmentId = fragment.getId();
        }
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar
if it is present.
        //
        getMenuInflater().inflate(R.menu.app_bar_items, menu);
        bindShopsAdapter();
        return true;
    }

    public void bindShopsAdapter() {
        Spinner spinner = findViewById(R.id.spnrShopList);
        String[] shopsToShow = new String[shops.size()];
        for (int i = 0; i < shops.size(); ++i) {
            shopsToShow[i] = shops.get(i).getName();
        }
        final ArrayAdapter<String> adapter = new
ArrayAdapter<>(this,
                android.R.layout.simple_spinner_item,
shopsToShow);

adapter.setDropDownViewResource(android.R.layout.simple_spinner_
dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent,
View view, int position, long id) {
                if (selectedShop.getId() !=
shops.get(position).getId()) {
                    selectedShop = shops.get(position);
                    selectedCategory = "";
//                    MainActivity.this.adapter.filter("shopId",
String.valueOf(selectedShop.getId()), false);
                    MainActivity.this.adapter.addAll(new
ArrayList<Item>()); // Clear item list if refreshed.
                    MainActivity.this.currentPage = 0;
```

```
MainActivity.this.fetchData.downloadItems(MainActivity.this.getC
urrentConfiguration());
                }
//                }
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent)
{}});
        if (shops.size() > 0 && selectedShop == null) {
            selectedShop = shops.get(0);
            adapter.notifyDataSetChanged();
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.action_choose_categories:
                // Show dialog with categories.
                //

MainActivity.this.getCategories(String.valueOf(selectedShop.getI
d()));
                break;
            case R.id.action_help:
                // Show dialog with help.
                //
                Dialog help = new Dialog(this);
                help.setContentView(R.layout.frame_help);
                help.show();
                break;
            case R.id.action_about:
                // Show dialog with about information.
                //
                Dialog about = new Dialog(this);
                about.setContentView(R.layout.frame_about);
                about.show();
                break;
        }
        return super.onOptionsItemSelected(item);
    }

    @Override
    protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        if (resultCode == 0) {
            finish();
        }
    }

    @Override
```

```
    public void onBackPressed() {
        if (getSupportFragmentManager().getBackStackEntryCount()
> 0) {
            getSupportFragmentManager().popBackStack();
        } else if (!doubleBackToExitPressedOnce) {
            this.doubleBackToExitPressedOnce = true;

            Toast.makeText(this, R.string.doubleBackExit,
Toast.LENGTH_SHORT).show();
            new Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    doubleBackToExitPressedOnce = false;
                }
            }, 2000);
        } else {
            setResult(0);
            finish();
        }
    }

    public void filterSelectedShops() {
        if (selectedShop == null) {
            adapter.returnToOld();
        } else {
            adapter.filter("shopId",
String.valueOf(selectedShop.getId()), false);
        }
    }

    public String getCurrentConfiguration() {
        String shopId;
        if (selectedShop == null) {
            shopId = "1";
        } else {
            shopId = String.valueOf(selectedShop.getId());
        }
        if (currentPage == 0) {
            currentPage = 1;
        }
        return new APIRequests.ItemsGETRequest(shopId,
selectedCategory, String.valueOf(currentPage)).getURL();
    }

    private View.OnClickListener loginLogoutListener = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (isLoggedIn()) {
                final AlertDialog.Builder alertDialog = new
AlertDialog.Builder(MainActivity.this);
                alertDialog.setTitle(R.string.loggin_out);
                alertDialog.setMessage(R.string.sure_or_not);
```

```
                alertDialog.setPositiveButton(R.string.okk,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface
dialog, int which) {
                                alertDialog.show();
SharedPrefsUtil.clearPrefs(MainActivity.this, Config.KEY_TOKEN);
                                Intent i = new
Intent(MainActivity.this, LoginActivity.class);
                                startActivityForResult(i, 0);
                                MainActivity.this.setResult(0);
                                MainActivity.this.finish();
                            }
                        });

                alertDialog.setNegativeButton(R.string.cancell,
                        new DialogInterface.OnClickListener() {
                            public void onClick(DialogInterface
dialog, int which) {
                                dialog.cancel();
                            }
                        });
                alertDialog.show();
            } else {
                Intent i = new Intent(MainActivity.this,
LoginActivity.class);
                startActivityForResult(i, 0);
                MainActivity.this.setResult(0);
                MainActivity.this.finish();
            }
        }
    };

    private View.OnClickListener addShopList = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final AlertDialog.Builder alertDialog = new
AlertDialog.Builder(MainActivity.this);
            alertDialog.setTitle(R.string.new_shoplist);
            alertDialog.setMessage(R.string.how_to_call);

            final EditText input = new
EditText(MainActivity.this);
            LinearLayout.LayoutParams lp = new
LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.MATCH_PARENT);
            input.setLayoutParams(lp);
            input.setSingleLine(true);
            alertDialog.setView(input);

            alertDialog.setPositiveButton(R.string.okk,
```

```
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {
                            String shopListName =
input.getText().toString();
                            if (TextUtils.isEmpty(shopListName))
{

Toast.makeText(MainActivity.this, R.string.please_enter_name,
Toast.LENGTH_LONG).show();
                            } else {

MainActivity.this.postAddShopList(shopListName);
                            }
                        }
                    });

            alertDialog.setNegativeButton(R.string.cancell,
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {
                            dialog.cancel();
                        }
                    });

            alertDialog.show();
        }

    };

    private void postAddShopList(final String name) {
        fetchData.beforeDownload();
        Response.Listener<String> respListener = new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (TextUtils.isEmpty(response)) {
                    AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
                    builder.setMessage(R.string.error_adding_sl)

.setNeutralButton(R.string.neutral_ok, null)
                            .create()
                            .show();
                } else {
                    ShopList shopList;
                    try {
                        shopList = ShopList.fromJSONObject(new
JSONObject(response));
                        if (shopList.getName().equals(name)) {
                            Toast.makeText(MainActivity.this,
getString(R.string.created) + name, Toast.LENGTH_LONG).show();
```

```
MainActivity.this.fetchData.execute(false);

MainActivity.this.fetchData.afterDownload();
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    MainActivity.this.fetchData.afterDownload();
                }
            }
        };

        Response.ErrorListener errListener = new
Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e(Config.TAG_VOLLEY_ERROR,
error.toString());
                Toast.makeText(MainActivity.this,
R.string.error_adding_sl, Toast.LENGTH_LONG).show();
            }
        };

        Map<String, String> namePayload = new HashMap<>();
        namePayload.put("name", name);
        final JSONObject jsonPayload =
JSONUtil.formPayload(namePayload);

        String userToken = SharedPrefsUtil.getStringPref(this,
Config.KEY_TOKEN);
        Map<String, String> headers = new HashMap<>();
        headers.put("Authorization", "Bearer " + userToken);

        APIRequests.RequestHandler rh =
APIRequests.formPOSTRequest(
                true,
                jsonPayload,
                headers,
                Config.URL_SHOPLIST,
                respListener,
                errListener,
                new WeakReference<>((Context) this)
        );

        rh.launch();
    }

    private void getCategories(String shopId) {
        fetchData.beforeDownload();
        final Response.Listener<String> respListener = new
Response.Listener<String>() {
            @Override
```

```
            public void onResponse(String response) {
                if (TextUtils.isEmpty(response)) {
                    AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);

builder.setMessage(R.string.error_loading_categories)

.setNeutralButton(R.string.neutral_ok, null)
                            .create()
                            .show();
                } else {
                    List<String> categs = new ArrayList<>();
                    try {
                        JSONArray ja = new JSONArray(response);
                        for (int i = 0; i < ja.length(); i++) {
                            categs.add((String) ja.get(i));
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                    MainActivity.this.categories = categs;
                    fetchData.afterDownload();

                    final String[] categories = new
String[MainActivity.this.categories.size() + 1];
                    categories[0] =
getString(R.string.all_categs);
                    for (int i = 1; i < categories.length; i++)
{
                        categories[i] =
MainActivity.this.categories.get(i - 1);
                    }
                    AlertDialog.Builder builder = new
AlertDialog.Builder(MainActivity.this);
                    builder.setTitle(R.string.select_category);
                    builder.setItems(categories, new
DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int pos) {
                            if
(categories[pos].equals(getString(R.string.all_categs))) {
                                selectedCategory = "";
                            } else {
                                selectedCategory =
categories[pos];
                            }
                            MainActivity.this.adapter.addAll(new
ArrayList<Item>()); // Clear item list if refreshed.
                            MainActivity.this.currentPage = 0;
                            fetchData.execute(true);

MainActivity.this.fetchData.afterDownload();
```

```
//
MainActivity.this.fetchData.downloadItems(MainActivity.this.getC
urrentConfiguration());
                        }
                    });
                    AlertDialog alert = builder.create();
                    alert.show();
                }
            }
        };

        Response.ErrorListener errListener = new
Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e(Config.TAG_VOLLEY_ERROR,
error.toString());
                Toast.makeText(MainActivity.this,
R.string.error_loading_categories, Toast.LENGTH_LONG).show();
            }
        };

        APIRequests.RequestHandler rh =
APIRequests.formGETRequest(
                Config.URL_SALES_SHOP + shopId + "/" +
Config.URL_CATEGORIES,
                null,
                respListener,
                errListener,
                new WeakReference<>((Activity)
MainActivity.this)
        );

        rh.launch();
    }
}
```

Код класу RegisterActivity:

```
public class RegisterActivity extends AppCompatActivity {

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

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        this.etPassword = findViewById(R.id.etRegPassword);
        this.etUsername = findViewById(R.id.etRegUsername);
        this.btnRegister = findViewById(R.id.btnRegister);
```

```
        btnRegister.setOnClickListener(registerListener);
    }

    private View.OnClickListener registerListener = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (!checkFields()) {
                return;
            }

            final String username =
etUsername.getText().toString();
            final String password =
etPassword.getText().toString();
            userData = new HashMap<>();
            userData.put(Config.KEY_USERNAME, username);
            userData.put(Config.KEY_PASSWORD, password);
            final JSONObject jsonPayload =
JSONUtil.formPayload(userData);

            Response.Listener<String> respListener = new
Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    if (TextUtils.isEmpty(response) ||
response.contains(Config.BAD_API_AUTH_RESPONSE)) {
                        AlertDialog.Builder builder = new
AlertDialog.Builder(RegisterActivity.this);
                        builder.setMessage(R.string.invalidUser)

.setNegativeButton(R.string.loginRetry, null)
                                .create()
                                .show();
                    } else {
                        Toast.makeText(getApplicationContext(),
getString(R.string.user_success_register) + " " + username,
Toast.LENGTH_LONG).show();
                        RegisterActivity.this.finish();
                    }
                }
            };

            Response.ErrorListener errListener = new
Response.ErrorListener() {
                @Override
                public void onErrorResponse(VolleyError error) {
                    Log.e(Config.TAG_VOLLEY_ERROR,
error.toString());

etUsername.setError(getString(R.string.invalidUser));
                }
            };
```

```
            APIRequests.RequestHandler rh =
APIRequests.formPOSTRequest(
                    true,
                    jsonPayload,
                    null,
                    Config.URL_REGISTER,
                    respListener,
                    errListener,
                    new WeakReference<>((Context)
RegisterActivity.this));
            rh.launch();
    private boolean checkFields() {
        boolean noErrors = true;
        if (TextUtils.isEmpty(etUsername.getText())) {

etUsername.setError(getString(R.string.emptyUsername));
            noErrors = false;
        }

        if (TextUtils.isEmpty(etPassword.getText())) {

etPassword.setError(getString(R.string.emptyPassword));
            noErrors = false;
        }
        return noErrors;
    }
}
```

Код класу ShopListActivity:

```
public class ShopListActivity extends AppCompatActivity {

    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;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_shop_list);

        Toolbar myToolbar = findViewById(R.id.toolbar);
        setSupportActionBar(myToolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowTitleEnabled(false);

        tvTotalPrice = findViewById(R.id.tvTotalPrice);
```

```
        tvShopListName = findViewById(R.id.tvShopListName);
        rvShopList = findViewById(R.id.rvShopList);
        swipeRefreshLayout =
findViewById(R.id.swipeSlContainer);
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            ((ProgressBar)
findViewById(R.id.pbSlLoading)).setProgress(0, true);
        } else {
            ((ProgressBar)
findViewById(R.id.pbSlLoading)).setProgress(0);
        }
        fetchData = new FetchData(this, swipeRefreshLayout);

        layoutManager = new LinearLayoutManager(this);
        rvShopList.setLayoutManager(layoutManager);

        adapter = new ItemAdapter(new ArrayList<Item>(), this);
        rvShopList.setAdapter(adapter);

        ShopList sl =
getIntent().getExtras().getParcelable(KEY_CURRENT_SHOPLIST);
        selectedShopList = sl;

fetchData.downloadCurrentShopList(selectedShopList.getId());

swipeRefreshLayout.setOnRefreshListener(refreshListener);
        btnAdd = findViewById(R.id.action_item_add);
        btnAdd.setOnClickListener(addCustomItem);

tvShopListName.setText(truncate(selectedShopList.getName(),
10));
    }

    SwipeRefreshLayout.OnRefreshListener refreshListener = new
SwipeRefreshLayout.OnRefreshListener() {
        @Override
        public void onRefresh() {
            ShopListActivity.this.adapter.addAll(new
ArrayList<Item>());
            layoutManager = new
LinearLayoutManager(ShopListActivity.this);
            rvShopList.setLayoutManager(layoutManager);

fetchData.downloadCurrentShopList(selectedShopList.getId());
        }
    };

    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.collapseActionView:
                return true;

            default:
```

```
                onBackPressed();
                return true;
        }
    }

    private View.OnClickListener addCustomItem = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            final AlertDialog.Builder alertDialog = new
AlertDialog.Builder(ShopListActivity.this);
            alertDialog.setTitle(R.string.add_custom_item);

alertDialog.setMessage(R.string.title_add_custom_item);


            final EditText input = new
EditText(ShopListActivity.this);
            LinearLayout.LayoutParams lp = new
LinearLayout.LayoutParams(
                    LinearLayout.LayoutParams.MATCH_PARENT,
                    LinearLayout.LayoutParams.MATCH_PARENT);
            input.setLayoutParams(lp);
            input.setSingleLine(true);
            alertDialog.setView(input);

            alertDialog.setPositiveButton(R.string.okk,
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {
                            String shopListName =
input.getText().toString();
                            if (TextUtils.isEmpty(shopListName))
{

Toast.makeText(ShopListActivity.this,
R.string.please_enter_name, Toast.LENGTH_LONG).show();
                            } else {

ShopListActivity.this.postItemAdd(shopListName);
                            }
                        }
                    });

            alertDialog.setNegativeButton(R.string.cancell,
                    new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface
dialog, int which) {
                            dialog.cancel();
                        }
                    });

            alertDialog.show();
```

```
        }
    };


    private void postItemAdd(final String name) {
        fetchData.beforeDownload();
        Response.Listener<String> respListener = new
Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                if (TextUtils.isEmpty(response)) {
                    AlertDialog.Builder builder = new
AlertDialog.Builder(ShopListActivity.this);
                    builder.setMessage(R.string.error_adding_sl)

.setNeutralButton(R.string.neutral_ok, null)
                            .create()
                            .show();
                } else {
                    Item item;
                    try {
                        item = Item.fromJSONObject(new
JSONObject(response));
                        if (item.getName().equals(name)) {

Toast.makeText(ShopListActivity.this,
getString(R.string.item_added) + " " + name,
Toast.LENGTH_LONG).show();

ShopListActivity.this.adapter.addAll(new ArrayList<Item>());
                            layoutManager = new
LinearLayoutManager(ShopListActivity.this);

rvShopList.setLayoutManager(layoutManager);

fetchData.downloadCurrentShopList(selectedShopList.getId());
                        }
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }

ShopListActivity.this.fetchData.afterDownload();
                }
            }
        };

        Response.ErrorListener errListener = new
Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Log.e(Config.TAG_VOLLEY_ERROR,
error.toString());
```

```
                Toast.makeText(ShopListActivity.this,
R.string.error_adding_sl, Toast.LENGTH_LONG).show();
            }
        };

        String userToken = SharedPrefsUtil.getStringPref(this,
Config.KEY_TOKEN);
        Map<String, String> headers = new HashMap<>();
        headers.put("Authorization", "Bearer " + userToken);

        APIRequests.RequestHandler rh =
APIRequests.formPOSTRequest(
                false,
                null,
                headers,
                new
APIRequests.ShopListPOSTRequest(String.valueOf(selectedShopList.
getId()), name).getAddItemURL(),
                respListener,
                errListener,
                new WeakReference<>((Context)
ShopListActivity.this)
        );

        rh.launch();
    }

    private String truncate(String str, int len) {
        if (str.length() > len) {
            return str.substring(0, len) + "...";
        } else {
            return str;
        }
    }
}
```

Код класу LoginActivity:

```
public class LoginActivity extends AppCompatActivity {

    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;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
```

```
        sharedPrefs = getSharedPreferences(Config.SH_PREFS_NAME,
Context.MODE_PRIVATE);
        if (!getUserToken().equals(Config.DEF_NO_TOKEN)) {
            showMainActivity();
        }
        etUsername = findViewById(R.id.etUsername);
        etPassword = findViewById(R.id.etPassword);
        tvRegister = findViewById(R.id.tvRegister);
        btnLogin = findViewById(R.id.btnLogin);
        tvHack = findViewById(R.id.tvHack);

        btnLogin.setOnClickListener(btnLoginListener);
        tvRegister.setOnClickListener(registerListener);
        tvHack.setOnClickListener(hackListener);
    }

    {
        btnLoginListener = new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!checkFields()) {
                    return;
                }
                final String username =
etUsername.getText().toString();
                final String password =
etPassword.getText().toString();
                userData = new HashMap<>();
                userData.put(Config.KEY_USERNAME, username);
                userData.put(Config.KEY_PASSWORD, password);
                final JSONObject jsonPayload =
JSONUtil.formPayload(userData);

                Response.Listener<String> respListener = new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        if (TextUtils.isEmpty(response) ||
response.contains(Config.BAD_API_AUTH_RESPONSE)) {
                            AlertDialog.Builder builder = new
AlertDialog.Builder(LoginActivity.this);

builder.setMessage(R.string.invalidUser)

.setNegativeButton(R.string.loginRetry, null)
                                    .create()
                                    .show();
                        } else {
                            setUserToken(response);
                            showMainActivity();
                        }
                    }
                };
```

```
                Response.ErrorListener errListener = new
Response.ErrorListener() {
                    @Override
                    public void onErrorResponse(VolleyError
error) {
                        Log.e(Config.TAG_VOLLEY_ERROR,
error.toString());
                        Toast.makeText(getApplicationContext(),
R.string.vLoginError, Toast.LENGTH_LONG).show();
                    }
                };

                APIRequests.RequestHandler rh =
APIRequests.formPOSTRequest(true, jsonPayload, null,
                        Config.URL_LOGIN, respListener,
errListener, new WeakReference<>((Context) LoginActivity.this));
private View.OnClickListener registerListener = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i = new Intent(LoginActivity.this,
RegisterActivity.class);
            startActivity(i);
        }
    };
    private View.OnClickListener hackListener = new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent i = new Intent(LoginActivity.this,
MainActivity.class);
            startActivityForResult(i, 0);
        }
    };
    private boolean checkFields() {
        boolean noErrors = true;
        if (TextUtils.isEmpty(etUsername.getText())) {

etUsername.setError(getString(R.string.emptyUsername));
            noErrors = false;
        }
        if (TextUtils.isEmpty(etPassword.getText())) {

etPassword.setError(getString(R.string.emptyPassword));
            noErrors = false;
        }
        return noErrors;
    }
    private String getUserToken() {
        return sharedPrefs.getString(Config.KEY_TOKEN,
Config.DEF_NO_TOKEN);
    }
```

```
    private void setUserToken(String userToken) {
        SharedPreferences.Editor e = sharedPrefs.edit();
        e.putString(Config.KEY_TOKEN, userToken);
        e.apply();
    }
    private void showMainActivity() {
        Intent i = new Intent(LoginActivity.this,
MainActivity.class);
        startActivityForResult(i, 0);
    }
    @Override
    protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        if (resultCode == 0) {
            finish();
        }
    }}
```

Додаток Г

Код парсера головного файла middleware:

```
class SlotPolicy(object):
    PER_DOMAIN = 'per_domain'
    SINGLE_SLOT = 'single_slot'
    SCRAPY_DEFAULT = 'scrapy_default'

    _known = {PER_DOMAIN, SINGLE_SLOT, SCRAPY_DEFAULT}


class SplashCookiesMiddleware(object):
    """
    This downloader middleware maintains cookiejars for Splash
requests.

    It gets cookies from 'cookies' field in Splash JSON
responses
    and sends current cookies in 'cookies' JSON POST argument
instead of
    sending them in http headers.

    It should process requests before SplashMiddleware, and
process responses
    after SplashMiddleware.
    """
    def __init__(self, debug=False):
        self.jars = defaultdict(CookieJar)
        self.debug = debug

    @classmethod
    def from_crawler(cls, crawler):
        return
cls(debug=crawler.settings.getbool('SPLASH_COOKIES_DEBUG'))

    def process_request(self, request, spider):
        """
        For Splash requests add 'cookies' key with current
        cookies to ``request.meta['splash']['args']`` and remove
cookie
        headers sent to Splash itself.
        """
        if 'splash' not in request.meta:
            return

        if request.meta.get('_splash_processed'):
            request.headers.pop('Cookie', None)
            return

        splash_options = request.meta['splash']

        splash_args = splash_options.setdefault('args', {})
```

```
        if 'cookies' in splash_args:  # cookies already set
            return
        if 'session_id' not in splash_options:
            return

        jar = self.jars[splash_options['session_id']]

        cookies = self._get_request_cookies(request)
        har_to_jar(jar, cookies)

        splash_args['cookies'] = jar_to_har(jar)
        self._debug_cookie(request, spider)

    def process_response(self, request, response, spider):
        """
        For Splash JSON responses add all cookies from
        'cookies' in a response to the cookiejar.
        """
        from scrapy_splash import SplashJsonResponse
        if not isinstance(response, SplashJsonResponse):
            return response

        if 'cookies' not in response.data:
            return response

        if 'splash' not in request.meta:
            return response

        if not request.meta.get('_splash_processed'):
            warnings.warn("SplashCookiesMiddleware requires
SplashMiddleware")
            return response

        splash_options = request.meta['splash']
        session_id = splash_options.get('new_session_id',

splash_options.get('session_id'))
        if session_id is None:
            return response

        jar = self.jars[session_id]
        request_cookies = splash_options['args'].get('cookies',
[])
        har_to_jar(jar, response.data['cookies'],
request_cookies)
        self._debug_set_cookie(response, spider)
        response.cookiejar = jar
        return response

    def _get_request_cookies(self, request):
        if isinstance(request.cookies, dict):
            return [
```

```
                {'name': k, 'value': v} for k, v in
request.cookies.items()
            ]
        return request.cookies or []
    def _debug_cookie(self, request, spider):
        if self.debug:
            cl = request.meta['splash']['args']['cookies']
            if cl:
                cookies = '\n'.join(
                    'Cookie: {}'.format(self._har_repr(c)) for c
in cl)
                msg = 'Sending cookies to:
{}\n{}'.format(request, cookies)
                logger.debug(msg, extra={'spider': spider})

    def _debug_set_cookie(self, response, spider):
        if self.debug:
            cl = response.data['cookies']
            if cl:
                cookies = '\n'.join(
                    'Set-Cookie: {}'.format(self._har_repr(c))
for c in cl)
                msg = 'Received cookies from:
{}\n{}'.format(response, cookies)
                logger.debug(msg, extra={'spider': spider})

    @staticmethod
    def _har_repr(har_cookie):
        return '{}={}'.format(har_cookie['name'],
har_cookie['value'])


class SplashDeduplicateArgsMiddleware(object):
    """
    Spider middleware which allows not to store duplicate Splash
argument
    values in request queue. It works together with
SplashMiddleware downloader
    middleware.
    """
    local_values_key = '_splash_local_values'

    def process_spider_output(self, response, result, spider):
        for el in result:
            if isinstance(el, scrapy.Request):
                yield self._process_request(el, spider)
            else:
                yield el

    def process_start_requests(self, start_requests, spider):
        if not hasattr(spider, 'state'):
            spider.state = {}
```

```
        spider.state.setdefault(self.local_values_key, {})  #
fingerprint => value dict

        for req in start_requests:
            yield self._process_request(req, spider)
    def _process_request(self, request, spider):
        """
        Replace requested meta['splash']['args'] values with
their fingerprints.
        This allows to store values only once in request queue,
which helps
        with disk queue size.
        Downloader middleware should restore the values from
fingerprints.
        """
        if 'splash' not in request.meta:
            return request
        if '_replaced_args' in request.meta['splash']:
            # don't process re-scheduled requests
            # XXX: does it work as expected?
            warnings.warn("Unexpected
request.meta['splash']['_replaced_args']")
            return request
        request.meta['splash']['_replaced_args'] = []
        cache_args = request.meta['splash'].get('cache_args',
[])
        args = request.meta['splash'].setdefault('args', {})
        for name in cache_args:
            if name not in args:
                continue
            value = args[name]
            fp = 'LOCAL+' + json_based_hash(value)
            spider.state[self.local_values_key][fp] = value
            args[name] = fp

request.meta['splash']['_replaced_args'].append(name)
        return request
class SplashMiddleware(object):
    """
    Scrapy downloader and spider middleware that passes requests
    through Splash when 'splash' Request.meta key is set.

    This middleware also works together with
SplashDeduplicateArgsMiddleware
    spider middleware to allow not to store duplicate Splash
argument values
    in request queue and not to send them multiple times to
Splash
    (the latter requires Splash 2.1+).
    """
    default_splash_url = 'http://127.0.0.1:8050'
    default_endpoint = "render.json"
    splash_extra_timeout = 5.0
```

```
    default_policy = SlotPolicy.PER_DOMAIN
    rescheduling_priority_adjust = +100
    retry_498_priority_adjust = +50
    remote_keys_key = '_splash_remote_keys'

    def __init__(self, crawler, splash_base_url, slot_policy,
log_400):
        self.crawler = crawler
        self.splash_base_url = splash_base_url
        self.slot_policy = slot_policy
        self.log_400 = log_400
        self.crawler.signals.connect(self.spider_opened,
signals.spider_opened)
    @classmethod
    def from_crawler(cls, crawler):
        splash_base_url = crawler.settings.get('SPLASH_URL',

cls.default_splash_url)
        log_400 = crawler.settings.getbool('SPLASH_LOG_400',
True)
        slot_policy = crawler.settings.get('SPLASH_SLOT_POLICY',
                                           cls.default_policy)
        if slot_policy not in SlotPolicy._known:
            raise NotConfigured("Incorrect slot policy: %r" %
slot_policy)
        return cls(crawler, splash_base_url, slot_policy,
log_400)
    def spider_opened(self, spider):
        if not hasattr(spider, 'state'):
            spider.state = {}
        # local fingerprint => key returned by splash
        spider.state.setdefault(self.remote_keys_key, {})
    @property
    def _argument_values(self):
        key = SplashDeduplicateArgsMiddleware.local_values_key
        return self.crawler.spider.state[key]
    @property
    def _remote_keys(self):
        return self.crawler.spider.state[self.remote_keys_key]
    def process_request(self, request, spider):
        if 'splash' not in request.meta:
            return
        if request.method not in {'GET', 'POST'}:
            logger.warning(
                "Currently only GET and POST requests are
supported by "
                "SplashMiddleware; %(request)s will be handled
without Splash",
                {'request': request},
                extra={'spider': spider}
            )
        if request.meta.get("_splash_processed"):
            # don't process the same request more than once
```

```
            return
        splash_options = request.meta['splash']
        request.meta['_splash_processed'] = True
        slot_policy = splash_options.get('slot_policy',
self.slot_policy)
        self._set_download_slot(request, request.meta,
slot_policy)
        args = splash_options.setdefault('args', {})

        if '_replaced_args' in splash_options:
            # restore arguments before sending request to the
downloader
            load_args = {}
            save_args = []
            local_arg_fingerprints = {}
            for name in splash_options['_replaced_args']:
                fp = args[name]
                # Use remote Splash argument cache: if Splash
key
                # for a value is known then don't send the value
to Splash;
                # if it is unknown then try to save the value on
server using
                # ``save_args``.
                if fp in self._remote_keys:
                    load_args[name] = self._remote_keys[fp]
                    del args[name]
                else:
                    save_args.append(name)
                    args[name] = self._argument_values[fp]

                local_arg_fingerprints[name] = fp

            if load_args:
                args['load_args'] = load_args
            if save_args:
                args['save_args'] = save_args
            splash_options['_local_arg_fingerprints'] =
local_arg_fingerprints

            del splash_options['_replaced_args']  # ??

        args.setdefault('url', request.url)
        if request.method == 'POST':
            args.setdefault('http_method', request.method)
            # XXX: non-UTF8 request bodies are not supported now
            args.setdefault('body', request.body.decode('utf8'))

        if not splash_options.get('dont_send_headers'):
            headers =
scrapy_headers_to_unicode_dict(request.headers)
            if headers:
                args.setdefault('headers', headers)
```

```
        body = json.dumps(args, ensure_ascii=False,
sort_keys=True, indent=4)
        # print(body)

        if 'timeout' in args:

            timeout_requested = float(args['timeout'])
            timeout_expected = timeout_requested +
self.splash_extra_timeout
            # no timeout means infinite timeout
            timeout_current =
request.meta.get('download_timeout', 1e6)
            if timeout_expected > timeout_current:
                request.meta['download_timeout'] =
timeout_expected
        endpoint = splash_options.setdefault('endpoint',
self.default_endpoint)
        splash_base_url = splash_options.get('splash_url',
self.splash_base_url)
        splash_url = urljoin(splash_base_url, endpoint)

        headers = Headers({'Content-Type': 'application/json'})
        headers.update(splash_options.get('splash_headers', {}))
        new_request = request.replace(
            url=splash_url,
            method='POST',
            body=body,
            headers=headers,
            priority=request.priority +
self.rescheduling_priority_adjust
        )
        self.crawler.stats.inc_value('splash/%s/request_count' %
endpoint)
        return new_request
    def process_response(self, request, response, spider):
        if not request.meta.get("_splash_processed"):
            return response
        splash_options = request.meta['splash']
        if not splash_options:
            return response
        # update stats
        endpoint = splash_options['endpoint']
        self.crawler.stats.inc_value(
            'splash/%s/response_count/%s' % (endpoint,
response.status)
        )
        # handle save_args/load_args
        self._process_x_splash_saved_arguments(request,
response)
        if get_splash_status(response) == 498:
            logger.debug("Got HTTP 498 response for {}; "
```

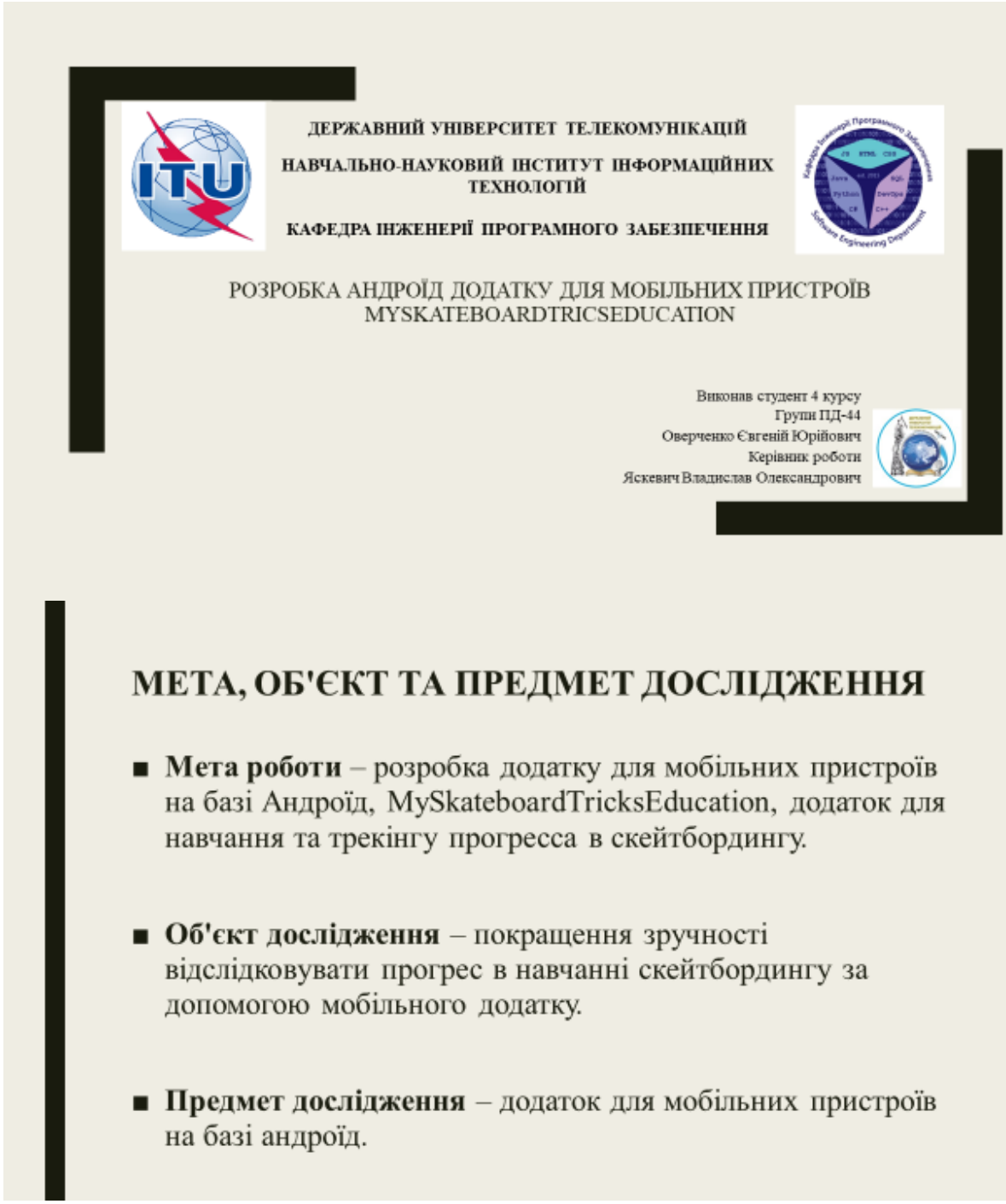
```
                            "sending arguments
again.".format(request),
                            extra={'spider': spider})
                return self._498_retry_request(request, response)
            if splash_options.get('dont_process_response', False):
                return response
            response = self._change_response_class(request,
response)
            if self.log_400 and get_splash_status(response) == 400:
                self._log_400(request, response, spider)
            return response
        def _change_response_class(self, request, response):
            from scrapy_splash import SplashResponse,
SplashTextResponse
            if not isinstance(response, (SplashResponse,
SplashTextResponse)):
                # create a custom Response subclass based on
response Content-Type
                # XXX: usually request is assigned to response only
when all
                # downloader middlewares are executed. Here it is
set earlier.
                # Does it have any negative consequences?
                respcls =
responsetypes.from_args(headers=response.headers)
                if isinstance(response, TextResponse) and respcls is
SplashResponse:
                    # Even if the headers say it's binary, it has
already
                    # been detected as a text response by scrapy
(for example
                    # because it was decoded successfully), so we
should not
                    # convert it to SplashResponse.
                    respcls = SplashTextResponse
                response = response.replace(cls=respcls,
request=request)
            return response
        def _log_400(self, request, response, spider):
            from scrapy_splash import SplashJsonResponse
            if isinstance(response, SplashJsonResponse):
                logger.warning(
                    "Bad request to Splash: %s" % response.data,
                    {'request': request},
                    extra={'spider': spider}
                )
        def _process_x_splash_saved_arguments(self, request,
response):
            """ Keep track of arguments saved by Splash. """
            saved_args = get_splash_headers(response).get(b'X-
Splash-Saved-Arguments')
            if not saved_args:
                return
```

```
        saved_args =
parse_x_splash_saved_arguments_header(saved_args)
        arg_fingerprints =
request.meta['splash']['_local_arg_fingerprints']
        for name, key in saved_args.items():
            fp = arg_fingerprints[name]
            self._remote_keys[fp] = key

    def _498_retry_request(self, request, response):
        """
        Return a retry request for HTTP 498 responses. HTTP 498
means
        load_args are not present on server; client should retry
the request
        with full argument values instead of their hashes.
        """
        meta = copy.deepcopy(request.meta)
        local_arg_fingerprints =
meta['splash']['_local_arg_fingerprints']
        args = meta['splash']['args']
        args.pop('load_args', None)
        args['save_args'] = list(local_arg_fingerprints.keys())
        for name, fp in local_arg_fingerprints.items():
            args[name] = self._argument_values[fp]
            # print('remote_keys before:', self._remote_keys)
            self._remote_keys.pop(fp, None)
            # print('remote_keys after:', self._remote_keys)
        body = json.dumps(args, ensure_ascii=False,
sort_keys=True, indent=4)
        # print(body)
        request = request.replace(
            meta=meta,
            body=body,

priority=request.priority+self.retry_498_priority_adjust
        )
        return request
    def _set_download_slot(self, request, meta, slot_policy):
        if slot_policy == SlotPolicy.PER_DOMAIN:
            # Use the same download slot to (sort of) respect
download
            # delays and concurrency options.
            meta['download_slot'] = self._get_slot_key(request)
        elif slot_policy == SlotPolicy.SINGLE_SLOT:
            # Use a single slot for all Splash requests
            meta['download_slot'] = '__splash__'
        elif slot_policy == SlotPolicy.SCRAPY_DEFAULT:
            # Use standard Scrapy concurrency setup
            pass
    def _get_slot_key(self, request_or_response):
        return self.crawler.engine.downloader._get_slot_key(
            request_or_response, None
        )
```

Додаток Д


ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
РОЗРОБКА АНДРОЇД ДОДАТКУ ДЛЯ МОБІЛЬНИХ ПРИСТРОЇВ MYSKATEBOARDTRICSEDUCATION
Виконав студент 4 курсу
Групи ПД-44
Оверченко Євгеній Юрійович
Керівник роботи
Яскевич Владислав Олександрович
МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ
Мета роботи – розробка додатку для мобільних пристроїв на базі Андроїд, MySkateboardTricksEducation, додаток для навчання та трекінгу прогресса в скейтбордингу.
Об'єкт дослідження – покращення зручності відслідковувати прогрес в навчанні скейтбордингу за допомогою мобільного додатку.
Предмет дослідження – додаток для мобільних пристроїв на базі андроїд.

ВИБІР МОВИ ПРОГРАМУВАННЯ

У веб-програмуванні зазвичай використовують

JavaScript - **на стороні клієнта**

PHP, Java, C ++, C, JavaScript, ASP.NET, Python – **на стороні сервера**

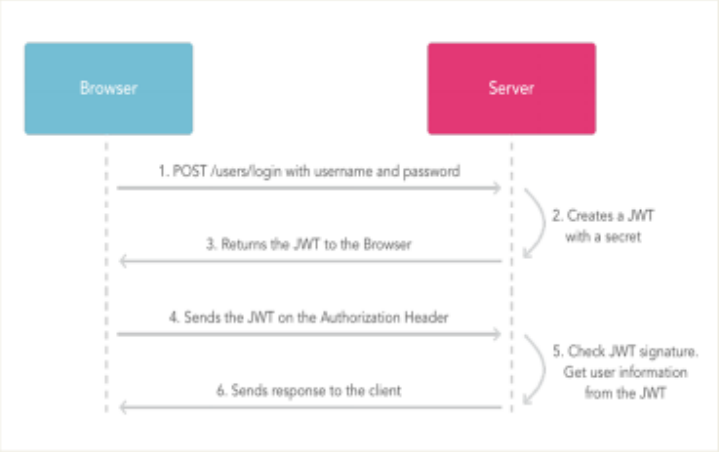
Jan 2020	Jan 2019	Change	Programming Language	Ratings	Change
1	1		Java	16.896%	-0.01%
2	2		C	15.773%	+2.44%
3	3		Python	9.704%	+1.41%
4	4		C++	5.574%	-2.58%
5	7	^	C#	5.349%	+2.07%
6	5	v	Visual Basic .NET	5.287%	-1.17%
7	6	v	JavaScript	2.451%	-0.85%
8	8		PHP	2.405%	-0.28%
9	15	^^	Swift	1.795%	+0.61%
10	9	v	SQL	1.504%	-0.77%

список десяти мов програмування за індексом TIOBE станом на січень 2020 року

Алгоритмічне та матиматечне рішення планування серверу

В даному випадку представлено програмне забезпечення на архітектурі клієнт-сервер

- **Клієнт відправляє дані браузеру, в даному випадку клієнт це Android-додаток;**
- **Сервер отримує дані, створює пакет, перевіряє дані, запаковує пакет та посилає клієнту;**
- **Клієнт отримує дані, розпаковує пакет та обробляє дані запаковує пакет з новими даними та відправляє серверу на перевірку;**
- **Сервер отримує пакет, розпаковує обробляє та перепаковує дані та відправляє клієнту.**



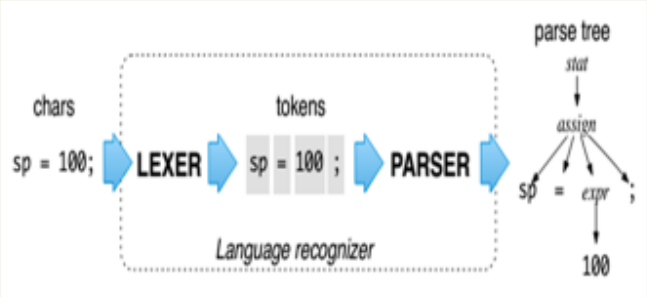
Алгоритмічне та матиматечне рішення парсеру

Парсити - збирати і систематизувати інформацію, розміщену на певних сайтах, за допомогою спеціальних програм, що автоматизують процес

Парсинг використовується для:

- Аналізу цінової політики
- Відстеження змін
- Наведення порядку на своєму сайті
- Наповнення карток трюків в інтернет-магазині
- Отримання баз потенційних клієнтів.

Алгоритм дії парсера

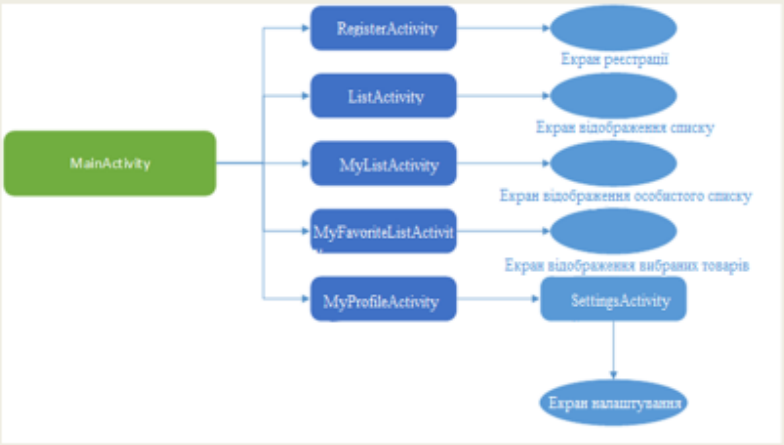


Алгоритмічне та матиматечне рішення клієнта

Виділено такі активності:

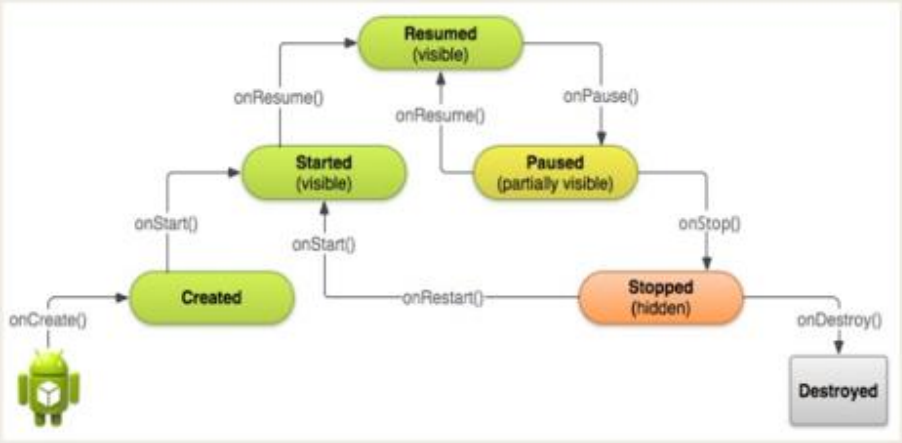
- RegisterActivity;
- ListActivity;
- MyListActivity;
- MyFavoriteListActivity;
- MyProfileActivity.

Ілюстрація того, як активності утворюють відгалуження у макеті та містять активності



Activity може існувати тільки в одному з трьох станів протягом тривалого періоду часу:

- **resumed (поновлення)**
- **paused (призупинено)**
- **stopped (зупинено).**



Ілюстрація життєвого циклу Activity, виражена у вигляді ступінчастої піраміди.

Як виглядає додаток?

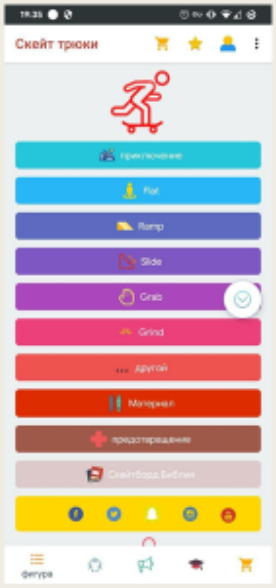
При вході в додаток нас зустрічає екран з привітанням, а після екран реєстрації

Реєстрація дає можливість роботи зі списками трюків, тому для входу потрібно увійти. Якщо ж клієнт вже зареєстрований в системі він може увійти, а опісля переглянути списки трюків.

Увійшовши в обліковий запис або вибравши режим без реєстрації, користувач потрапляє на головний екран додатка.

Екран списку розділів

На даному знімку екрану зображено зображено список розділів Зручний список «Улюблені» з кнопкою «+» дозволяє користувачу додати в обрані супермаркет. Також далі, з будь-якої активності користувач може викликати розділ help в меню додатку, де вказана інструкція користування даним продуктом. Також, користувач має можливість переглянути розділ about, де написана загальна інформація про програму і про розробника, і при довгому натисканні на елемент кнопку користувач отримає можливість дізнатися функціонал додатку, що допоможе йому краще розбиратися в роботі програми.



Екран каталогу нових розділів

Перейшовши в каталог розділу можемо побачити новий добавлений матеріал у вкладці «New»

Екран каталогу Roll to Dice

Відкривши цю вкладку можна зручно зіграти в класичну гру серед скейтерів - SKATE

ВИСНОВКИ

I. Представлено етапи розробки інформаційної системи та покрокове виконання.

II. Розробка розпочата із так званого сервера де виділено бізнес-логіку у вигляді сутностей класів, покроково описано розробку пакетів та класів даного модуля.

III. Наступним етапом розробки був парсер, який відповідно до вимог повинен був розшукувати інформацію та витягувати з веб-сторінок, покрокова реалізація представлена вище у розділі та головним лістингом у додатку Д.

IV. Останнім підпунктом розробка клієнта, який був розроблений за допомогою так званих екранів активностей котрі взаємодіють між собою, а також між сервером, отримуючи дані та відображаючи їх.

V. Дана система протестована за допомогою ручного тестування.

Дякую за увагу!