Міністерство освіти і науки України

Криворізький національний університет

Кафедра моделювання та програмного забезпечення

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

На тему: «Розробка програмних засобів мобільних телефонів для реєстрації груп користувачів»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,**

Проектував (виконав) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Мартинюк А.Ю./

Керівник дипломного проекту (роботи) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Котов І.А./

Нормативний контролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Хорошенька Л.В./

Завідуючий кафедрою \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Азарян А.А./

Кривий Ріг

2013

Міністерство освіти і науки України

Криворізький національний університет

Кафедра моделювання та програмного забезпечення

Спеціальність: ..програмне забезпечення автоматизованих систем…

Затверджую

Зав. кафедри..Азарян А.А. .. .. .. ..

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ЗАВДАННЯ

на дипломний проект

студент групи…ПЗАС-09-1…………………………………………………………………\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема: «Розробка програмних засобів мобільних телефонів для реєстрації груп користувачів»……\_\_\_\_\_\_\_\_ \_\_\_\_\_\_ …\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

затверджено наказом по КНУ №\_\_\_\_\_\_\_\_\_\_\_\_ від “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_ 2013р.

2. Термін подання студентом закінченого проекту “\_\_\_\_”\_\_\_\_\_\_\_\_\_\_ 2013р.

3. Вихідні дані по проекту:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст пояснювальної записки (перелік питань, що їх треба розробити)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_н.р\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Перелік графічного демонстраційного матеріалу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_назви плакатів\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Календарний план:

|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Етапи роботи** | **Термін виконання** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Дата видачі завдання: “\_\_\_\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_р.

Завдання видав:

Керівник дипломного проекту \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Котов І.А.

Завдання отримав:

Студент-дипломник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Мартинюк А.Ю.

**РЕФЕРАТ**

Дипломна робота «Розробка програмних засобів мобільних телефонів для реєстрації груп користувачів» присвячена збору інформації від різних груп людей за допомогою засобів мобільних пристроїв.

В цій роботі розглядається проблема отримання даних від користувачів за допомогою пристроїв, які є мобільними та можуть бути технічно придатними для збору даних від інших пристроїв. В якості рішення проблеми мобільності пропонується використання мобільних пристроїв.

В роботі описуються основні завдання, які необхідно реалізувати для збору даних від користувачів мобільних пристроїв.

В роботі проводиться аналіз методів збору інформації, на основі чого проводиться вибір оптимального методу, який буде зручно використовувати на мобільних пристроях. В якості методу збору інформації пропонується використовувати метод анкетування.

Для мобільних пристроїв проводиться аналіз існуючих рішень, що дозволяють проводити збір інформації від користувачів. Демонструються можливості та недоліки, які притаманні різним засобам в аспекті саме збору інформації.

На основі аналізу існуючих рішень та виявлених недоліків розглядається питання про необхідність створення нової розробки для задоволення проблеми збору даних від користувачів мобільних пристроїв.

Для аналізу накопичених даних пропонується використання додаткового програмного забезпечення, яке також розглядається в даній роботі. Для аналізу даних в додатковому програмному забезпеченні пропонується кількісний метод аналізу, що дає можливість порівняльного аналізу різних величин.

В роботі проводиться розробка структури програмної системи за допомогою створення функціональної схеми роботи системи, проводиться розробка та опис структури бази даних, проводиться розробка та опис алгоритмів роботи системи та проектування інтерфейсу.

В дипломній роботі також розглянуто питання охорони праці та безпеки життєдіяльності.

ЗМІСТ

Зм

Арк.

№ документа

Підпис

Дата

Аркуш

4

H

КНУ.ПД.6.050103.13.13

Розроб.

Мартинюк

Перевір.

Котов

Н. Контр.

Хорошенька

Розробка програмних засобів мобільних телефонів для реєстрації груп користувачів

Літера

Аркушів

92

ПЗАС-09-1

Затверд.

*Азарян*

[Вступ 5](#_Toc358321631)

[1 Аналіз проблеми реєстрації груп користувачів за допомогою мобільних пристроїв 6](#_Toc358321632)

[1.1 Критичний аналіз літературних джерел з даної теми 6](#_Toc358321633)

[1.2 Сучасний стан та актуальність досліджуваної проблеми 11](#_Toc358321634)

[1.3 Аналіз існуючих мобільних платформ 14](#_Toc358321635)

[1.3.1 Аналіз платформи Java ME 14](#_Toc358321636)

[1.3.2 Аналіз платформи iOS 16](#_Toc358321637)

[1.3.3 Аналіз платформи Android 17](#_Toc358321638)

[1.4 Задача створення системи реєстрації груп користувачів засобами мобільних телефонів 19](#_Toc358321639)

[2 Розробка функціональної схеми і алгоритму роботи системи 21](#_Toc358321640)

[2.1 Розробка і детальний опис функціональної схеми 21](#_Toc358321641)

[2.2 Розробка і детальний опис алгоритму роботи системи 29](#_Toc358321642)

[3 Розробка структури бази даних і програмного забезпечення 34](#_Toc358321643)

[3.1 Розробка структури БД 34](#_Toc358321644)

[3.2 Розробка програмного забезпечення 38](#_Toc358321645)

[3.2.1 Проектування інтерфейсу програми 38](#_Toc358321646)

[3.2.2 Опис програми 42](#_Toc358321647)

[3.2.3 Посібник користувача 45](#_Toc358321648)

[4 Охорона праці та безпека життєдіяльності 51](#_Toc358321649)

[4.1 Охорона праці 51](#_Toc358321650)

[4.2 Безпека життєдіяльності 58](#_Toc358321651)

[Висновок 63](#_Toc358321652)

[Список використаних джерел 64](#_Toc358321653)

# ВСТУП

Дипломна робота присвячена темі «Розробка програмних засобів мобільних телефонів для реєстрації груп користувачів».

Часто перед початком проведення різних суспільних заходів при знайомстві з аудиторією необхідно отримати певні дані від аудиторії. Цими даними можуть бути як базовий рівень знань аудиторії, під який буде підлаштовуватися курс, якість матеріалу, що був викладений на конференції, дані про осіб для певного аналізу та інші дані.

В даній роботі буде вирішуватися проблема з отриманням цих даних. Наразі це можна зробити, запропонувавши групі осіб відповісти на запитання в усній формі, що є не дуже зручним, та інформація для аналізу не може бути отримана від великої кількості користувачів. Отримання інформації в письмовій формі, що вирішує проблему з великою кількістю людей, але для аналізу це також не пристосована інформація, її потрібно аналізувати вручну або переводити в цифровий вид. Сайти для створення тестів, збору інформації дозволяють швидко зібрати інформацію та опрацювати її, але сторонні ресурси не завжди гарантують анонімність та часто є випадки, коли немає доступу до Інтернету або пристроїв з гарною роздільною здатність екрану для зручного перегляду певних сайтів.

При зборі інформації інколи приходиться збирати дані, які є типовими і можуть бути підготовлені завчасно, після чого тільки необхідно провести процедуру передачі цих даних.

Сьогодні багато людей користуються мобільними пристроями і кількість цих людей постійно зростає. На даному етапі мобільні пристрої мають гарні технічні показники та гарні програмні платформи, що дозволяють створювати різні додатки для них. Саме тому мобільна платформа є вигідним рішенням для вирішення питання мобільного збору інформації.

В роботі проводиться створення програмної системи, що б дозволила вирішити певні проблеми, які пов’язані зі збором даних. Також в роботі будуть розглянуті додаткові можливості, які представлені аналізом цих даних.

# АНАЛІЗ ПРОБЛЕМИ РЕЄСТРАЦІЇ ГРУП КОРИСТУВАЧІВ ЗА ДОПОМОГОЮ МОБІЛЬНИХ ПРИСТРОЇВ

## Критичний аналіз літературних джерел з даної теми

Для автоматизованого збору інформації використовуються різні системи збору даних. В даній роботі буде йти мова про розробку програмної системи для збору даних, тому розглянемо поняття системи збору даних.

Система збору даних - комплекс засобів, що використовуються для роботи разом з персональним комп’ютером або спеціалізованою ЕОМ та виконуючий автоматизований збір інформації з аналогових або цифрових джерел сигналів, а також обробку, накопичення і передачу даних.

Перед описом структури та функцій системи для збору даних розглянемо методи для збору інформації.

Вибір методу збору інформації є важливим питанням, так як від нього залежить ефективність, швидкість та якість збору та отриманої інформації.

Достатньо розповсюдженою формою опитувань є інтерв'ю. Найхарактерніша його особливість як специфічного виду опитування полягає в тому, що інтерв'юер (той, хто опитує) і респондент (той, кого опитують) зводяться обличчям в обличчя, що інформація, яка цікавить дослідника, міститься у відповідях індивіда на задане йому в усній формі запитаннях.[4]

Інтерв'ю — це метод одержання необхідної інформації шляхом безпосередньої цілеспрямованої бесіди інтерв'юера з респондентом. Напрям бесіди визначається тією проблемою, яка цікавить інтерв'юера і є предметом дослідження.

Існують різні види інтерв’ю, розглянемо найпоширеніші з них:

* вільне інтерв'ю, коли, як правило, немає плану і завчасно сформульованих запитань;
* глибинне інтерв'ю має за мету отримати інформацію, яка засвідчує не лише наявність того чи іншого соціального факту, явища, але й пояснює причини появи даних фактів, явищ;
* сфокусоване спрямоване інтерв'ю — вивчення громадської думки відносно конкретної події, факту, ситуації;
* стандартизоване (формалізоване) інтерв'ю, коли формулювання запитань, їх порядок, кількість і перелік можливих альтернативних відповідей, їх кодування і форма запису передбачаються заздалегідь і суворо фіксуються в своїй одноманітності.[4]

Розглянемо ще один метод збору інформації, що має назву анкетування. Серед розповсюджених методів опитування респондентів важливе місце займає метод анкетного опитування. Надзвичайна популярність цього методу пояснюється різноманітністю і якістю соціологічної інформації, яку можна одержати за його допомогою. Цей метод засновується на висловлюваннях окремих осіб і проводиться з метою виявлення найтонших нюансів в думках респондентів.

Анкета, як правило, починається з преамбули — звернення до респондента. В ньому вказується, хто проводить дослідження, описується мета дослідження, характер використання результатів, підкреслюється анонімний характер, спосіб заповнення анкети, а також висловлюється подяка (вона можлива і в кінці анкети) за участь в анкетуванні. Далі йде основна частина анкети, яка містить блоки запитань до опитуваних, і третя частина — паспортичка, в якій представлені відомості про опитуваних (стать, вік, місце мешкання, соціальний стан і т. д.).

За способом розповсюдження анкет опитування поділяються на:

* роздаткові - коли анкетер особисто вручає анкету і чекає, поки вона заповнюється, і тут же одержує її — очне роздаткове опитування, або одержує заповнену анкету через декілька днів — заочне роздавальне опитування;
* поштове - анкета висипається і одержується поштою за попередньою домовленістю або без неї;
* пресове - анкету пропонують заповнити читачеві газети чи журналу і надіслати в редакцію;

За типом дослідницьких завдань опитування бувають:

* стандартизоване — націлене на одержання статистичної інформації;
* сфокусоване — збираються дані за умов конкретної ситуації;
* глибинне — спрямоване на одержання пошукової інформації.

За рівнем компетентності респондентів розрізняють:

* масове опитування (думка неспеціалістів з тої чи іншої теми);
* масове опитування у співробітництві з дослідником (передбачає інформаційну допомогу респонденту з боку анкетера в осмисленні ситуації, що аналізується);
* симптоматичне опитування (достатнє знання у респондента загальної інформації без глибокого осмислення цілей і завдань дослідження);
* експертне опитування (опитування спеціалістів з проблеми, що вивчається).

Розглянемо наступний метод збору інформації - спостереження. Спостереження як метод збирання інформації не можна віднести до специфічних методів. В соціологічно-природничому симбіозі спостереження можна в першому наближенні визначити як планомірне цілеспрямоване сприйняття явищ, результати якого в тій або іншій формі фіксуються дослідником і потім перевіряються.

При цьому збирається і фіксується за допомогою технічних приладів (кінокамера, фотоапарат, магнітофон, телевізійна та інша техніка) лише та інформація, яка може бути використана для опису, а потім і пояснення проблемної ситуації, яка досліджується.

Розрізняють також відкрите спостереження, яке характеризується тим, що членам досліджуваної групи факт спостереження за ними відомий, від групи він не приховується, і спостереження інкогніто, коли члени спостережної групи не підозрюють, що за ними ведеться спостереження.

Як основний метод збирання первинної інформації метод спостереження є досить ефективним в монографічних дослідженнях, тобто дослідженнях окремого випадку, а також в дослідженнях, які не вимагають великого обсягу вибірки.

Розглянемо методи аналізу отриманої інформації, так як аналіз є важливою частиною системи збору інформації, так як дозволяє отримати узагальнені дані про зібрану нами інформацію.

Розрізняють наступні методи аналізу інформації:

* традиційний аналіз (класичний). Тлумачення документа через з'ясування основних думок та ідей конкретного тексту. Оцінку його змісту згідно з політичними, моральними або естетичними критеріями;
* контент-аналіз (кількісний). Аналіз змісту масової сукупності текстів з використанням стандартизованих вимірювальних, статистичних процедур з метою отримання їх об'єктивних кількісних характеристик. Його сутність полягає у перекладі в кількісні показники текстової інформації.

Найпоширенішими є традиційний (класичний) і формалізований (якісно-кількісний) метод аналізу документів. Саме під традиційним, класичним аналізом розуміється вся багатоманітність розумових операцій, спрямованих на інтеграцію відомостей, що містяться в документі з певної точки зору, застосованого дослідником в кожному конкретному випадку. Традиційний аналіз документів дає можливість дати свою інтерпретацію змісту документа, проникнути в сутність явища, що вивчається, виявити логічні зв'язки і протиріччя між ними, оцінити ці явища і факти з певних дослідницьких позицій.[5]

Бажання позбутися суб'єктивності традиційного аналізу обумовило появу принципово нового, формалізованого метода аналізу документів, який одержав назву «контент-аналіз». Контент-аналіз, або науковий аналіз змісту тексту (документа), — це метод дослідження, який застосовується в різних гуманітарних дисциплінах. Але розвиток цього методу переважно пов'язаний з соціологічними дослідженнями ЗМІ. З часом метод контент-аналізу став застосовуватися і при вивченні інших галузей соціальної реальності, іншого типу документів, зокрема, невербальних, іконографічних (портрети, фотографії і т. д.), а також відповідей на відкриті запитання соціологічної анкети. Цей метод дуже часто використовується різними спецслужбами: до 80 відсотків таємної інформації здобуваються за його допомогою і зараз.

Суть цього методу зводиться до того, щоб знайти такі ознаки, риси, властивості документа (наприклад, частота вживання певних термінів), які з необхідністю віддзеркалювали б певні суттєві сторони змісту. Тоді зміст документа стає вимірюваним, доступним точним обчислювальним операціям. Разом з тим обмеженість контент-аналізу полягає в тому, що далеко не все багатство змісту документа може бути виміряне за допомогою формальних (кількісних) показників.

В контент-аналізі передбачається своя вибірка. Масив документів, об'єднаних загальною ознакою, представляє генеральну сукупність, яка має бути вивчена. Якщо стоїть завдання провести контент-аналіз кампаній по виборах президента України, то в межі генеральної сукупності треба включити всі документи, які опубліковані у зв'язку з виборами з моменту початку кампанії до її завершення.[5]

Для формування вибіркової сукупності документів для контент-аналізу зазвичай використовують суцільний і випадковий відбір. Щодо застосування скерованого типу відбору, то він виключається через відсутність даних про розподіл в генеральній сукупності документів.

Плідним виявляється при аналізі текстів діяльнісний (проблемний) підхід. В цьому випадку весь текст розглядається як опис конкретної проблемної ситуації, в якій є низка суб'єктів та відносини між ними. При формалізованому аналізі документів всебічно розглядають саму діяльність, а також виділяють її суб'єкти, цілі та мотиви вчинків, що здійснюються ними; обставини, причини, що породили потребу в тій чи іншій діяльності (бездіяльність - це теж вид діяльності); об'єкт її напрямки.

Після розглянутих методів збору інформації та аналізу можна виділити саме ті, як найкраще піддаються автоматизації та можуть бути використаними при розробці програмних засобів для реєстрації груп користувачів.

Серед методів збору інформації можна виділити той, який найбільше розповсюджений та може бути легко автоматизованим. Цим методом є анкетування. На відмінну від спостереження та інтерв’ю не потрібна присутність людини, що проводитиме різні опитування. Опитування можна представити в електронному вигляді і дати можливість користувачам самим вибирати різні дані.

Серед методів аналізу можна вибрати кількісний метод аналізу. Цей метод, на відміну від класичного методу аналізу, легко автоматизувати так як він оперує з числовими даними. Класичний метод потребував би створення системи аналізу змісту тексту і видачі певних тверджень на основі цього аналізу.

## Сучасний стан та актуальність досліджуваної проблеми

Аналізуючи сучасний стан досліджуваної проблеми, а саме проблему збору інформації від різних користувачів за допомогою мобільних пристроїв, необхідно проаналізувати стандартні можливості мобільних пристроїв, за допомогою яких в певній мірі можна вирішити дану проблему.

До стандартних можливостей мобільних пристроїв можна віднести SMS повідомлення, електронну пошту, дзвінки.

Розглянемо технологію SMS. Служба коротких повідомлень (SMS) — послуга обміну (передачі і прийому) короткими текстовими повідомленнями в телекомунікаційних мережах, доступна для більшості мобільних телефонів та інших комунікаційних пристроїв, таких як пейджер, модем, КПК, або, навіть, настільний комп'ютер (за допомогою функцій програмного забезпечення). Для реалізації послуга повинна підтримуватись оператором зв'язку, комунікаційним пристроєм та програмним забезпеченням комунікаційного пристрою. Для випадку мобільного зв'язку для користування послугою потрібна SIM-карта. Технологія SMS підтримується мобільними мережами GSM, NMT, D-AMPS, CDMA, UMTS.

Максимальна стандартна довжина одного SMS-повідомлення становить до 160 знаків латиницею або 70 кирилицею. Довші повідомлення розбиваються на кілька повідомлень. Оплата здійснюється за кожне повідомлення окремо згідно з умовами договору (тарифного плану, контракту).

Розглянемо наступний стандартний інструмент мобільних пристроїв, який наразі є в багатьох пристроях, а саме електронну пошту. Електронна пошта (e-mail) — популярний сервіс в Інтернеті, що робить можливим обмін даними будь-якого змісту (текстові документи, аудіо-відео файли, архіви, програми).

Електронною поштою можна надсилати не лише текстові повідомлення, але й документи, графіку, аудіо -, відео файли, програми тощо. Через електронну пошту можна отримати послуги інших сервісних мереж.

Електронна пошта — типовий сервіс відкладеного зчитування (off-line). Після відправлення повідомлення, як правило, у вигляді звичайного тексту, адресат отримує його на свій комп'ютер через деякий період часу, і знайомиться з ним, коли йому буде зручно.

Електронна пошта схожа на звичайну пошту. Звичайний лист складається із конверта, на якому зазначена адреса отримувача і стоять штампи поштових відділень шляху слідування, та вмісту — власне листа. Електронний лист складається із заголовків, які містять службову інформацію (про автора листа, отримувача, шлях проходження листа), які служать, умовно кажучи, конвертом, та власне вміст самого листа. За аналогією зі звичайним листом, відповідним методом можна внести в електронний лист інформацію якого-небудь іншого роду, наприклад, фотографію тощо. Як і у звичайному листі можна поставити свій підпис. Звичайний лист може не дійти до адресата або дійти з запізненням, — аналогічно і електронний лист. Звичайний лист доволі дешевий, а електронна пошта — найдешевший вид зв'язку.

Отже, електронна пошта повторює переваги (простоту, дешевизну, можливість пересилання нетекстової інформації, можливість підписати і зашифрувати лист) та недоліки (негарантований час пересилки, можливість доступу для третіх осіб під час пересилки, неінтерактивність) звичайної пошти. Проте у них є і суттєві відмінності. Вартість пересилки звичайної пошти у значній мірі залежить від того, куди вона повинна бути доставлена, її розміру та типу. У електронної пошти такої залежності або немає, або вона досить невідчутна. Електронний лист можна шифрувати та підписувати більш надійніше та зручніше, ніж лист на папері — для останнього, власне, взагалі не існує загальноприйнятих засобів шифровки. Швидкість доставки електронних листів набагато вища, ніж паперових, та мінімальний час проходження незрівнянно менший. Загалом в залежності від розміру листа та швидкості каналу зв'язку доставка електронного листа триває в середньому від кількох секунд до кількох хвилин. Щоправда, можуть бути затримки через збої в поштових серверах.

Телефоні дзвінки дозволяють збирати інформацію в виді опитування або інтерв’ю, але для швидкості та зручності збору це не підходить. Потрібно заздалегідь обговорювати час та тривалість бесіди з кожним респондентом.

Після аналізу стандартних засобів можна виділити їх недоліки:

* всі стандартні методи є платними, тому їх не бажано використовувати для масового збору інформації. Також вони є платними в таких випадках як sms також і для респондентів.
* стандартні методи не надають зручного інтерфейсу для збору інформації. Респондент має в вигляді тексту власноручно набирати відповіді на запитання, що може знизити аудиторію, яка захоче прийняти участь в зборі інформації.
* отримана інформація за допомогою стандартних методів не може бути проаналізована зовнішніми програмами, а тільки користувачем.

Розглянемо програму для мобільних пристроїв - TwedgeCE. TWedgeCE є універсальною програмою для збору даних для мобільних пристроїв і PDA (рисунок 1.1), що працюють під управлінням Windows Mobile®.

Програма дозволяє отримати дані із зовнішніх джерел, таких як сканери штрих-кодів, датчиків, електронних ваг, і направити їх у різні програми.

Після завантаження та встановлення, потрібно налаштувати TWedgeCE. Конфігурація відбувається в простому текстовому файлі, що спрощує використання ідентичних налаштувань на численних пристроях.

Якщо вихідні дані можуть бути прочитані з допомогою даної програми, то це дуже зручний засіб так як може бути інтегрований з будь-якою програмою.

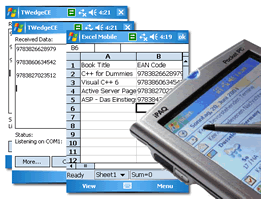


Рисунок 1.1 – Програма TwedgeCE

Ціна однієї ліцензії становить 71$.

Недоліки. Платформа Windows Mobile®, на якій зараз не випускаються пристрої, є застарілою, висока ціна за одну копію програми.

### 1.3 Аналіз існуючих мобільних платформ

### 1.3.1 Аналіз платформи Java ME

Java Platform, Micro Edition (Java ME) — підмножина платформи Java для пристроїв, з обмеженими ресурсами, наприклад: стільникових телефонів, кишенькових комп'ютерів, ресиверів цифрового телебачення, програвачів дисків Bluray тощо.

Java ME розроблена під керівництвом Sun Microsystems і є заміною схожої технології — PersonalJava. Спочатку специфікація розроблялася в рамках JCP (Java Community Process) як JSR 68. Пізніше її варіанти розвинулися в окремі JSR. Sun надає зразок реалізації (англ. reference implementation) специфікації, але спочатку не надавала безплатної реалізації середовища виконання (англ. runtime environment) Java ME для мобільних пристроїв. 22 грудня 2006 сирцевий код Java ME був випущений під ліцензією GNU General Public License, проект отримав назву phoneME.

Відмінними особливостями пристроїв з обмеженими ресурсами є обмежена обчислювальна потужність, обмежений обсяг пам'яті, малий розмір дисплея, живлення від портативної батареї, а також низькошвидкісні і недостатньо надійні комунікаційні можливості. Типовий сучасний мобільний телефон містить всередині 32-розрядний RISC-процесор з тактовою частотою 150-250 МГц, має об'єм оперативної пам'яті близько 1-2 Мб, кольоровий дисплей розміром 2 дюйми і має можливість з'єднання з Інтернетом за допомогою GPRS або EDGE зі швидкістю до 474 кбіт/с, яке при цьому фундаментально ненадійно, оскільки швидкість передачі даних може несподівано впасти, або з'єднання може бути взагалі повністю втрачено.

Java ME специфікує дві базові конфігурації, які визначають вимоги до віртуальної машини (обмеження набору допустимих інструкцій тощо), а також мінімальний набір базових класів: CLDC (Connected Limited Device Configuration - конфігурація пристрою з обмеженими ресурсами та комунікаційними можливостями) та CDC (Connected Device Configuration - конфігурація пристрою з нормальними ресурсами та комунікаційними можливостями).

Java ME також визначає кілька так званих профілів (англ. profiles), які доповнюють і розширюють згадані вище конфігурації, зокрема визначають модель програми, можливості графічного інтерфейсу, а також комунікаційні функції (наприклад доступ до Інтернету) тощо.

В даний час найпоширенішою конфігурацією є CLDC, для якої розроблений профіль MIDP (Mobile Information Device Profile - профіль для мобільного пристрою з інформаційними функціями). Програми, написані для цього профілю, називаються мідлети (англ. MIDlet). Іншим популярним профілем для CLDC є DoJa, розроблений фірмою NTT DoCoMo для її власного сервісу iMode. iMode вельми поширений в Японії, і в меншій мірі в Європі і на Далекому Сході.

Конфігурація CLDC успішно використовується в більшості сучасних мобільних телефонів і портативних органайзерів. За даними компанії Sun Microsystems до кінця 2004 у світі було випущено більше 579 мільйонів мобільних пристроїв з підтримкою цієї конфігурації Java. Це робить Java ME панівною технологією Java у світі. Обсяги виробництва мобільних телефонів значно перевищують кількість інших комп'ютерних пристроїв, здатних виконувати програми на Java (наприклад, персональних комп'ютерів).

На жаль, J2ME дуже обмежений у своїх можливостях і «габаритах» кінцевого продукту. Це стосується далеко не всіх телефонів, але якщо писати максимально крос-девайсовий продукт, всі ці обмеження варто враховувати.

Перелічимо основні з них:

* розмір скомпільованого JAR-файлу - на 20% java-пристроїв обмеження до 128Кб;
* доступ до файлової системи - лише за підтримки PIM API;
* робота з зображеннями - в основному підтримується тільки PNG-8 і PNG-24 без можливості накладення напівпрозорих областей на фонову графіку;
* сховище даних додатки - на деяких телефонах максимальний обсяг збережених даних становить всього 3Кб.

### 1.3.2 Аналіз платформи iOS

iOS (до 24 червня 2010 року - iPhone OS) - мобільна операційна система, що розробляється і випускається американською компанією Apple. Була випущена в 2007 році; спочатку - для iPhone і iPod touch, пізніше - для таких пристроїв, як iPad і Apple TV. На відміну від Windows Phone і Google Android, випускається тільки для пристроїв, вироблених фірмою Apple.

Інтерфейс користувача iOS заснований на концепції прямого маніпулювання з використанням жестів мультитач. Елементи управління інтерфейсом складаються з повзунків, перемикачів і кнопок.

iOS розроблена на основі Mac OS X і використовує той же набір основних компонентів Darwin, сумісний зі стандартом POSIX.

У iOS є чотири шари абстрагування: шар Core OS, шар Core Services, шар Media Layer, і шар Cocoa Touch.

Для поточної версії операційної системи (iOS 6.1.3) виділяється 1,4-2 Гб флеш-пам'яті пристрою для системного розділу і приблизно 800 Мб вільного місця (варіюється залежно від моделі).

Станом на 19 травня 2013 магазин додатків App Store містить більше 900 тис. додатків для iOS, які всі разом були завантажені більше 50 мільярдів разів.

Інші програми можуть бути розроблені за допомогою Xcode для Mac, Codea для iPad, і опубліковані в App Store - онлайн-магазині, який поставляється з самим iPhone / iPod touch / iPad, починаючи з версії iPhone OS 2.0, і є найбільшим магазином мобільних додатків (на травень 2012 року - більше 650 тис. додатків для iOS, більше 225 тис. додатків спеціально для iPad).

Основні конкуренти iOS:Android (Google), Windows Phone (Microsoft), BlackBerry OS (Research In Motion).

Деякі виробники пристроїв також випускають свою власну операційну систему для мобільних пристроїв.

iOS займає близько 14% ринка мобільних пристроїв, тому вона дуже поширена.

### 1.3.3 Аналіз платформи Android

Android — операційна система і платформа для мобільних телефонів та планшетних комп'ютерів, створена компанією Google на базі ядра Linux. Підтримується альянсом Open Handset Alliance (OHA).

Хоча Android базується на ядрі Лінукс, він стоїть дещо осторонь Лінукс-спільноти та Лінукс-інфраструктури. Базовим елементом цієї операційної системи є реалізація Dalvik віртуальної машини Java, і все програмне забезпечення і застосування спираються на цю реалізацію Java.

У 75% смартфонів, проданих у третьому кварталі 2012 року, була встановлена ​​операційна система Android.

Додатки під операційну систему Android є програмами в нестандартному байт-коді для віртуальної машини Dalvik, для яких був розроблений формат настановних пакетів. Apk. Для роботи над додатками доступно безліч бібліотек:

* bionic (бібліотека стандартних функцій, несумісна з libc);
* libc (стандартна системна бібліотека мови С);
* мультимедійні бібліотеки на базі PacketVideo OpenCORE (підтримують такі формати, як MPEG4, H.264, MP3, AAC, AMR, JPG і PNG);
* SGL (движок двомірної графіки);
* OpenGL ES 1.0 ES 2.0 (движок тривимірної графіки);
* Surface Manager (забезпечує для додатків доступ до 2D/3D);
* WebKit (готовий движок для Web-браузера; обробляє HTML, JavaScript);
* FreeType (движок обробки шрифтів);
* SQLite (легковага СУБД, доступна для всіх додатків);
* SSL (протокол, що забезпечує безпечну передачу даних по мережі).

У порівнянні з звичайними додатками Linux, додатки Android підкоряються додатковим правилам:

* Content Providers - обмін даними між додатками;
* Resource Manager - доступ до таких ресурсів, як файли XML, PNG, JPEG;
* Notification Manager - доступ до рядку стану;
* Activity Manager - управління активними додатками.

Google пропонує для вільного скачування інструментарій для розробки (Software Development Kit), який призначений для x86-машин під операційними системами Linux, Mac OS X (10.4.8 або вище), Windows XP, Windows Vista і Windows 7. Для розробки потрібно JDK 5 або більш новий.

Розробку додатків для Android можна вести на мові Java (не нижче Java 1.5). Існує плагін для Eclipse - Android Development Tools (ADT), призначений для Eclipse версій 3.3-3.7. Також існує плагін для IntelliJ IDEA, який полегшує розробку Android-додатків, і для середовища розробки NetBeans IDE, який, починаючи з версії NetBeans 7.0, перестав бути експериментальним, хоч поки і не є офіційним. Крім того, існує Motodev Studio for Android - комплексне середовище розробки на базі Eclipse, що дозволяє працювати безпосередньо з Google SDK.

У 2009 році на додаток до ADT був опублікований Android Native Development Kit (NDK) - пакет інструментаріїв і бібліотек, що дозволяє вести розробку додатків мовою С/С++. NDK рекомендується використовувати для розробки ділянок коду, критичних до швидкості.

Додатки для операційної системи Android розповсюджуються за допомогою магазину додатків Android market.

22 жовтня 2008 Google оголосила про відкриття онлайн-магазину додатків для ОС Android - Android Market. За угодою, розробники отримують 70% прибутку, оператори стільникового зв'язку - 30%. У лютому 2009 року для розробників із США і Великобританії з'явилася можливість брати плату за свої додатки в Android Market. Компанія Sony Ericsson перша запустила власний канал в онлайн-магазині додатків Android Market. У ньому представлені програми та ігри, які рекомендовані компанією.

На грудень 2011 року, з моменту створення Android Market, було завантажено 10 млрд. додатків.

У березні 2012 року компанія Google об'єднала мультимедійні сервіси «Книги», «Android Market», «Музика» та інші в єдиний сервіс Google Play. Інтернет-магазин Google Play працює в 190 країнах, налічує більше 700 тисяч додатків, а за час роботи сервісу набралося близько 25 мільярдів скачувань.

## 1.4 Задача створення системи реєстрації груп користувачів засобами мобільних телефонів

Метою даного дипломного проекту є створення програмних засобів для реєстрації груп користувачів. Засоби повинні бути написані для популярних платформ та бути мобільними.

При створені засобів для реєстрації груп користувачів має бути використана мобільна платформа, що дозволить користуватися нею будь-де.

Завдання, які має виконувати розроблена система:

* + мобільна платформа;
  + зручний інтерфейс для збору даних;
  + низька ціна використання та бескоштоність для респондентів;
  + можливість збору інформації з використанням запитань різного типу;
  + збір інформації на пристрої з тестами;
  + можливість передачі запитань іншому пристрою та прийом відповідей;
  + анонімність збору даних;
  + збереження даних в базі даних;
  + аналіз отриманих даних;
  + можливість створення тестів на ПК та передачі на мобільний пристрій;
  + можливість передачі відповідей з мобільного пристрою на ПК.

Програмні засоби мають використовуватися в будь-якому місці без доступу до Інтернету.

При аналізі вбудованих можливостей мобільних пристроїв було виявлено, що вони не відповідають всім умовам. Наприклад, технологія смс не має зручного інтерфейсу для проведення опитувань, вона взагалі для цього не призначена. Також ця технологія не підтримує анонімність збору даних. Ще одним недоліком смс технології є висока ціна для використання для респондента.

Технологія e-mail також має ряд недоліків, які дуже схожі на недоліки смс-технології. Електронна пошта не має зручного інтерфейсу для проведення опитувань, технологія не підтримує анонімність збору даних.

Телефоні дзвінки також мають ряд недоліків при зборі інформації від великої групи користувачів. Тривалість проведення опитування буде дуже великою, так як не можна одночасно збирати дані від різних користувачів. Також цей метод не підтримує збір даних за допомогою анкетування, єдиний можливий варіант – це інтерв’ю.

Необхідно визначитися з мобільною платформою для розробки програмного забезпечення. Потрібно визначити, яким вимогам має відповідати платформа для реалізації програмних засобів для реєстрації груп користувачів. Приведемо перелік вимог до мобільної платформи:

* можливість зберігання даних в базі даних;
* наявність API для доступу до бездротових засобів комунікації Bluetooth або Wi-Fi;
* наявність зручних засобів для розробки додатків для мобільної платформи;
* можливість створення зручного та динамічного інтерфейсу для користувача;
* платформа має бути популярна для покриття великої частини користувачів мобільних пристроїв.

Беручи до уваги наведені вище вимоги до мобільних платформ, можна визначити, що технологія Java ME найменш підходить так як є її реалізації, які покривають технічні вимоги програмного забезпечення, але різні виробники можуть реалізовувати різні види API тому програма в рамках однієї платформи не буде підтримуватися багатьма пристроями.

Дві інші платформи, а саме iOS та Android повністю підтримують технічні вимоги програмного забезпечення. Так як Android займає близько 75% ринку мобільних пристроїв, а iOS тільки близько 14%, тому буде вибрана саме перша платформа.

Таким чином була вибрана широкорозповсюджена платформа. В якості технології зв’язку між пристроями виберемо технологію Bluetooth так як вона підтримується багатьма пристроями і на других платформах, що є плюсом при розширені використовуваних платформ.

# РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ І АЛГОРИТМУ РОБОТИ СИСТЕМИ

## Розробка і детальний опис функціональної схеми

Розробимо функціональну схему для системи реєстрації груп користувачів. Дані опитування мають імпортуватися в базу даних на мобільному пристрої. Ці дані будемо зберігати в зовнішньому файлі. Для зручної модифікації та читання файлу представимо його в форматі XML. Текст файлу з одним опитуванням представлений нижче.

<?xml version="1.0" standalone="yes"?>

<NewDataSet>

<Table>

<test.\_id>7</test.\_id>

<test.title>Programing</test.title>

<create\_date>2013-05-21T23:14:42+03:00</create\_date>

<question.\_id>10</question.\_id>

<id\_test>7</id\_test>

<question.title>Який тип даних відсутній в Java?

</question.title>

<type>Singlechoose</type>

<variants>char~double~unsigned int</variants>

</Table>

</NewDataSet>

Функціональна схема імпорту представлена на рисунку 2.1.



Рисунок 2.1 – функціональна схема імпорту опитувань

Розглянемо процедуру проходження опитування. Для проходження опитування необхідно вибрати з бази даних з таблиці опитування необхідне опитування і перейти до складання його на даному пристрої або передати на віддалений пристрій. На віддаленому пристрої можна пройти опитування та результат передати на серверний пристрій. При цьому після отримання даних про складання локально або від віддаленого пристрою дані зберігаються в базі даних. Функціональна схема проходження опитувань представлена на рисунку 2.2.



Рисунок 2.2 – Функціональна схема складання опитувань

Для виведення даних з бази даних використаємо зовнішній файл в форматі XML в якому будемо зберігати відповіді на опитування з бази даних. Приклад вихідного XML файлу представлений нижче.

<?xml version="1.0" standalone="yes"?>

<DataSet>

<Row>

<\_id>11</\_id>

<id\_question>13</id\_question>

<answer\_text>Google Chroome</answer\_text>

</Row>

<Row>

<\_id>12</\_id>

<id\_question>14</id\_question>

<answer\_text>HTTP, FTP </answer\_text>

</Row>

</DataSet>

Функціональна схема збереження експортування відповідей наведена на рисунку 2.3.



Рисунок 2.3 – Функціональна схема експорту даних в файл

Тепер об’єднаємо всі функціональні частини в одну функціональну схему. Дана схема буде відображати роботу програмного засобу мобільних телефонів реєстрації груп користувачів. На схемі показана процедура імпорту опитувань, які зберігаються у зовнішньому файлі, процедура проходження опитувань та процедура експорту відповідей для подальшого аналізу. При чому проходження опитувань можливе як на локальному пристрої так і на віддаленому. Процедура проходження залишається такою самою в обох випадках, тільки при проходженні на віддаленому пристрої опитування отримується від серверу, а не з бази даних і відповіді також відсилаються на сервер, а не зберігаються в локальній базі даних.

Функціональна схема роботи програми для реєстрації груп користувачів представлена на рисунку 2.4.



Рисунок 2.4 - Функціональна схема роботи програми для реєстрації груп користувачів

Для зручності створення та аналізу опитувань розробимо додаткову програму для персонального комп’ютера. Перевагами цієї програми перед мобільною версією є можливість використання клавіатури для зручного вводу даних та великого монітору для зручного представлення даних. Розробимо функціональну схему для програми для персонального комп’ютера. Програма має підтримувати створення опитувань, запитань, зберігання їх в базі даних, а також можливість експорту даних в файли XML та імпорту даних з файлів XML. Вихідний файл має використовуватися мобільним пристроєм, а дані для імпорту братись з вихідного файлу XML мобільного пристрою. Дана програма буде використовувати свою локальну базу даних для збереження інформації про опитування та відповіді на них. Дана програма буде використовуватися в зв’язку з основною програмою для мобільних пристроїв збору інформації від користувачів, тому функціональна схема має проектуватися з можливістю об’єднання з існуючим рішенням для мобільних пристроїв.

Функціональна схема програми для створення та аналізу опитувань представлена на рисунку 2.5.



Рисунок 2.5 - Функціональна схема програми для створення та аналізу опитувань

Об’єднаємо функціональні схеми програм для мобільного пристрою та для персонального комп’ютера в одну функціональну схему. Об’єднання проведемо на основі спільних блоків двох схем. Спільними блоками є блоки «Опитування XML» та «Відповіді XML», що містять зовнішні для обох схем дані. Дані для блоку «Опитування XML» створюються в програми для персонального комп’ютера та є вихідними для цієї версії програми, а вхідними ці дані є для основної версії програми для мобільних пристроїв. Дані для блоку «Відповіді XML» створюються основною версією програми для мобільних пристроїв в процесі проходження опитувань. Ці дані імпортуються в програму для персонального комп’ютера для подальшого аналізу більш зручним способом чим за допомогою мобільних пристроїв.

Функціональна схема двох версій програм представлена на рисунку 2.6.



Рисунок 2.6 – Функціональна схема системи реєстрації груп користувачів

Проведемо опис функціональної схеми. Базовою одиницею для збору інформації від користувачів буде опитування. Кожне опитування буде містити запитання з варіантами відповідей або без них.

Для створення опитування необхідні дані від користувача. Такими даними є назва опитування. Потім користувач заповнює опитування додаючи в нього запитання. Для додавання запитання також необхідна інформація від користувача, яка має містити текст запитання, тип та, при необхідності, варіанти відповідей.

Для проходження опитування його необхідно експортувати на мобільний пристрій. Після експорту необхідно вибрати опитування і можна починати проходити його. Всі результати опитувань зберігаються в базі даних на мобільному пристрої.

Для аналізу опитувань необхідно експортувати відповіді з мобільного пристрою в програму.

Проведемо опис схеми по блоках:

* блок «Дані користувача» – блок включає дані, які надходять від користувача програми. Це такі дані як назва опитування, назви запитань та варіанти відповідей для запитань, вибір користувача для певного запитання;
* блок «Створення опитування» - включає процедуру для створення пустого опитування, що не містить запитань та його редагування;
* блок «Створення запитання» - відповідає за створення запитань та додавання їх до опитувань. Також відповідає за редагування запитань. При створенні запитань можна вибирати тип запитання – запитання з одним варіантом відповіді, запитання з декількома варіантами відповідей;
* блок «Опитування PC» - містить таблицю з бази даних для зберігання опитувань;
* блок «Запитання PC» - містить таблицю бази даних для зберігання запитань;
* блок «Відповіді PC» - містить таблицю бази даних для зберігання відповідей на запитання;
* блок «Опитування XML» - містить зовнішні дані, що зберігаються у файлі для експорту даних з бази даних з персонального комп’ютера до бази даних мобільного пристрою;
* блок «Опитування Mobile» - містить таблицю з бази даних для зберігання опитувань;
* блок «Вибір опитування» - включає процедуру для вибору опитування на мобільному пристрої;
* блок «Відправлення віддаленому клієнту» - містить процедуру по відправленню опитування на віддалений пристрій. Дія цієї процедури заключається в тому, що пристрій починає прослуховування клієнтів за допомогою технології Bluetooth і при підключені клієнта йому відсилаються дані опитування;
* блок «Дані від віддаленого серверу» - містить процедуру прийому даних від серверу. Для прийому даних необхідного включити процедуру пошуку активного серверу з опитуваннями за допомогою технології Bluetooth і при підключені сервер відправляє дані на даний пристрій;
* блок «Проходження опитування» - містить процедуру проходження опитувань. Дані для цього блоку можуть бути отримані від віддаленого серверу або вибрані на поточному пристрої. Під час проходження опитувань користувач може обирати один або декілька з представлених варіантів відповідей в залежності від типу запитання;
* блок «Відповіді Mobile» - містить таблицю бази даних для збереження результатів проходження опитувань;
* блок «Відповіді XML» - зовнішні дані, що містяться в файлі для передачі результатів опитувань з мобільного пристроюю на комп’ютер.

На основі створеної функціональної схеми можна приступати до розробки алгоритмів роботи блоків цієї схеми. Схема містить структуру для двох окремих програм – основної, що використовується безпосередньо для проведення опитувань за допомогою мобільних пристроїв, та допоміжної програми, що обслуговує основну програму та дозволяє в зручному вигляді створювати опитування та аналізувати відповіді. Так як створюватимуться дві програми необхідно описати алгоритми роботи для основної мобільної версії програми та для допоміжної програми.

## Розробка і детальний опис алгоритму роботи системи

Розробимо алгоритм для створення нового опитування. Спочатку користувач має ввести назву опитування. Після введення назви потрібно вибрати її зі списку існуючих опитувань і перейти до створення запитань.

Для створення запитань необхідно вказати текст запитання, вибрити його тип. Типів запитань може бути три: просте запитання з текстовою відповіддю, запитання з одним варіантом вибору, запитання з багатьма варіантами вибору. Якщо запитання просте, то процедура створення завершується, інакше – потрібно ввести варіанти відповідей. Після створення опитування воно зберігається в базі даних. Алгоритм створення опитування показаний на рисунку 2.7.



Рисунок 2.7 – Алгоритм створення опитування

Розробимо алгоритм експортування опитування. Для експортування опитування необхідно вибрати назву опитування, яке необхідно експортувати. Потім з бази даних зчитуються всі запитання, які відносяться до вибраного опитування. Вибірка зберігається в файл, ім’я якого вводить користувач, в форматі XML.

Алгоритм експорту зображений на рисунку 2.8.



Рисунок 2.8 - Алгоритм експортування опитування

Розробимо алгоритм імпортування опитування. Для імпортування опитування необхідно вибрати файл з даними про опитування. Після чого потрібно розібрати вміст файлу і зробити запити до бази даних для додавання записів, що були прочитані з вхідного файлу. Для збереження опитувань використовується файл формату XML. Це дозволяє швидко реалізувати алгоритм розбору даних, так як бібліотеки або класи для роботи з XML файлами є в багатьох мовах програмування. Також формат є текстовим, що дозволяє редагувати його в будь-якому текстовому редакторі.

Алгоритм імпортування наведено на рисунку 2.9.



Рисунок 2.9 - Алгоритм імпортування опитування

Розробимо алгоритм проходження опитування на локальному пристрої. Для проходження опитування на локальному пристрої необхідно вибрати опитування, після чого необхідно вибрати дію «Почати опитування». Користувач отримає список запитань, відповівши на які відповіді заносяться до бази даних.

Алгоритм опитування на локальному пристрої наведений на рисунку 2.10.



Рисунок 2.10 - Алгоритм проходження опитування на локальному пристрої

Розробимо алгоритм проходження опитування на віддаленому пристрої. Для проходження опитування на віддаленому пристрої необхідно вибрати опитування, потім потрібно вибрати дію «Поділитися» на пристрої на якому є опитування. Пристрій буде очікувати з’єднання, після чого відішле питання віддаленому пристрою. Опитування відбувається на віддаленому пристрої, після чого відсилається на сервер від якого прийняв опитування. Дані опитування заносяться до бази даних.

Алгоритм проходження опитування на віддаленому пристрої наведений на рисунку 2.11.



Рисунок 2.11 - Алгоритм опитування на віддаленому пристрої

Розробимо алгоритм експортування відповідей. Для експортування відповідей необхідно вибрати опитування, відповіді на яке необхідно експортувати. Після чого необхідно вибрати дію «Експортувати дані». Дані будуть експортовані на зовнішній накопичувач мобільного пристрою. Після цього дані можна імпортувати в базу даних на комп’ютері.

Алгоритм експортування відповідей наведений на рисунку 2.12.



Рисунок 2.12 - Алгоритм експортування відповідей

Розробимо алгоритм аналізу відповідей. При наявності зібраних даних від користувачів можна проводити їх аналіз, робити певні висновки. Для цього необхідно вибрати опитування, потім вибрати запитання, відповіді на які будуть аналізуватися і вибрати дію «Аналіз даних». В залежності від типу питання буде показаний графік відповідного типу.

Алгоритм аналізу відповідей представлений на рисунку 2.13.



Рисунок 2.13 – Алгоритм аналізу відповідей

Після проектування алгоритмві роботи системи можна розпочинати проектувати базу даних.

# РОЗРОБКА СТРУКТУРИ БАЗИ ДАНИХ І ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## Розробка структури БД

При створені бази даних програмної системи необхідно вибрати систему керування базами даних, які будуть використовуватися. Для рішення даної задачі гарним вибором буде локальна база даних. В версії бази для десктопної версії буде вибрана MS Accsess, в версії для мобільної платформи доступна тільки версія SQLite, тому буде вибрана саме вона.

Система керування базами даних (СКБД) — комп'ютерна програма чи комплекс програм, що забезпечує користувачам можливість створення, збереження, оновлення, пошук інформації та контролю доступу в базах даних.

Можливості СКБД:

* дозволяється створювати БД (здійснюється за допомогою мови визначення даних DDL (Data Definition Language))
* дозволяється додавання, оновлення, видалення та читання інформації з БД (за допомогою мови маніпулювання даними DML, яку часто називають мовою запитів)

Можна надавати контрольований доступ до БД за допомогою:

* системи забезпечення захисту, яка запобігає несанкціонованому доступу до БД;
* системи керування паралельною роботою прикладних програм, яка контролює процеси спільного доступу до БД;
* система відновлення — дозволяє відновлювати БД до попереднього несуперечливого стану, що був порушений в результаті збою апаратного або програмного забезпечення.

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовуються спільно, та призначені для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації (записи даних) та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин: збереженої інформації та системи управління нею.

Розглянемо базу даних для програми реєстрації груп користувачів. База даних буде складатися з 5 таблиць: опитування, запитання, тип запитання, відповіді, користувач.

Розглянемо структуру таблиці «Опитування». Структура представлена в таблиці 3.1. У таблицю входить інформація про опитування і час їх створення.

Таблиця 3.1 – Дані про опитування.

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип | Описання | Ключ |
| \_id | Integer Autoincrement | Числове ключове поле, первинний ключ | PK |
| title | Text | Назва тесту |  |
| create\_date | Datetime | Дата та час створення |  |

Розглянемо структуру таблиці «Запитання». Структура представлена в таблиці 3.2. У таблицю входить інформація про запитання.

Таблиця 3.2 – Дані про запитання.

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип | Описання | Ключ |
| \_id | Integer Autoincrement | Числове ключове поле, первинний ключ | PK |
| id\_test | Integer | Ключ для ідентифікації належності до опитування | FK |
| title | Text | Текст запитання |  |
| id\_type | Integer | Тип запитання |  |
| variants | Text | Варіанти відповідей |  |

Розглянемо структуру таблиці «Тип запитання». Структура представлена в таблиці 3.3. У таблицю входить інформація про типи запитань.

Таблиця 3.3 – Дані про типи запитань.

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип | Описання | Ключ |
| \_id | Integer Autoincrement | Числове ключове поле, первинний ключ | PK |
| title | Text | Назва типу запитання |  |
| description | Text | Описання типу запитання |  |

Розглянемо структуру таблиці «Відповіді». Структура представлена в таблиці 3.4. У таблицю входить інформація про відповіді на запитання.

Таблиця 3.4 – Дані про відповіді.

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип | Описання | Ключ |
| \_id | Integer Autoincrement | Числове ключове поле, первинний ключ | PK |
| id\_question | Integer | Ключ для ідентифікації належності до запитання | FK |
| id\_user | Integer | Ключ для ідентифікації користувача | FK |
| answer\_text | Text | Текст відповіді |  |

Розглянемо структуру таблиці «Користувачі». Структура представлена в таблиці 3.5. У таблицю входить інформація про користувача та мобільний пристрій.

Таблиця 3.5 – Дані про користувачів.

|  |  |  |  |
| --- | --- | --- | --- |
| Назва поля | Тип | Описання | Ключ |
| \_id | Integer Autoincrement | Ключове поле, первинний ключ | PK |
| nickname | Text | Нікнейм користувача |  |
| phone | Text | Дані мобільного пристрою |  |

На основі розроблених таблиць проведемо нормалізацію бази даних. Для цього необхідно виділити зв’язки, які будуть між таблицями та методи збереження цілісності. При оновлені даних виберемо каскадне оновлення, при видалені – каскадне видалення. Таким чином в базі не будуть зберігатися не потрібні дані, які не мають зв’язків з головною таблицею, тобто база не буде надлишковою.

Нормалізація бази даних представлена на рисунку 3.1.



Рисунок 3.1 – Нормалізація бази даних

Допоміжна програма для основної мобільної версії програми буде також використовувати цю саму структуру бази даних.

Нормалізація бази даних графічно демонструє структуру базу даних, а саме структуру таблиць та зв’язки між ними, та допомагає краще зрозуміти структуру бази даних. В нашій базі ми створили 5 таблиць та встановили зв’язки між ними за допомогою первинних та вторинних індексів. Для збереження цілісності було обрано каскадне оновлення при оновлені даних, при видалені - каскадне видалення.

## Розробка програмного забезпечення

## Проектування інтерфейсу програми

Створимо головну форму, на якій буде відображатися список опитувань, кнопки імпорту опитування та з’єднання з іншим пристроєм для отримання опитування. В списку опитувань відображатиметься іконка тесту або іконка за замовчуванням, якщо для опитування не було встановлено іконку. Також в списку опитувань має відображатися назва опитування та час створення опитування.

Вид форми представлений на рисунку 3.2.

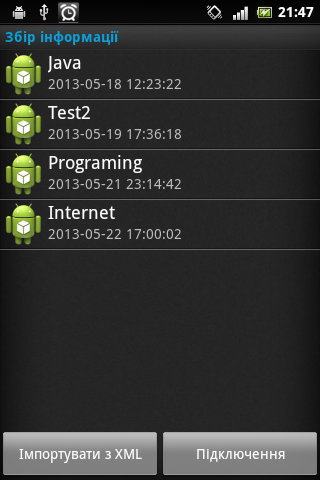


Рисунок 3.2 – Головна форма

Створимо форму для відображення можливих дій для з тестами. На цій формі мають відображатися кнопки для роздачі тесту по мережі, можливості відповісти на запитання на локальному пристрої, експортувати дані відповідей, видалення тесту. Кнопки «Роздати», «Експортувати в XML» та «Видалити» не повинні створювати нові вікна. Для повідомлення користувача про виконану дію повинно з’являтися спливаюче повідомлення з відповідним текстом. Кнопка «Пройти опитування» повинна створювати вікно для проходження опитування.

Вид форми представлений на рисунку 3.3.

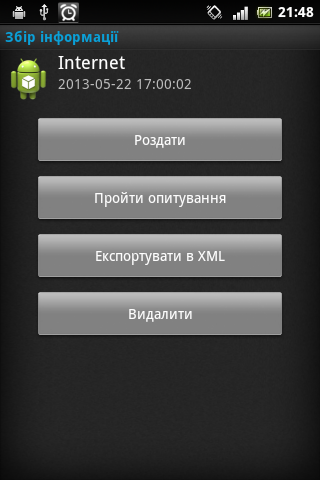


Рисунок 3.3 – Форма дій з опитуванням

Створимо форму для проведення опитування. Вона має містити тексти запитань, варіанти відповідей, поля для вводу даних.

Вид форми представлений на рисунку 3.4.

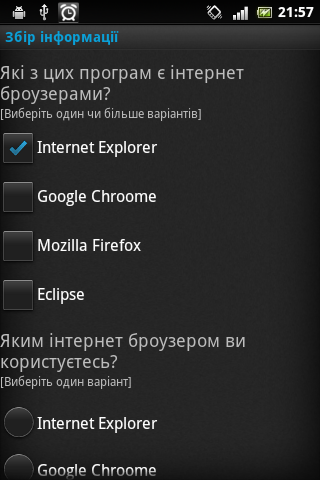


Рисунок 3.4 – Форма проходження опитувань

Розробимо інтерфейс для допоміжної програми, що буде працювати на персональному комп’ютері. Необхідно створити форму для створення запитань. Так як створення та аналіз будуть найчастіше використовуватися, то можна добавити їх на головну форму на різні вкладки. Форма для створення тестів буде включати 2 таблиці та кнопки для маніпулювання записами. Вид форми створення тестів та запитань наведено на рисунку 3.5.

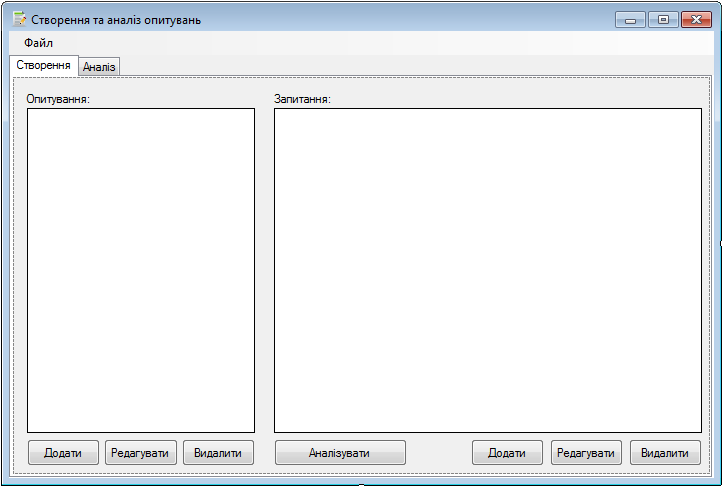


Рисунок 3.5 - Вид форми створення тестів та запитань

Для додавання та редагування тестів створимо додаткову форму з полем введення назви. Вид форми наведено на рисунку 3.6.

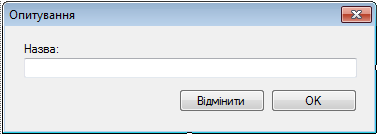


Рисунок 3.6 - Вид форми створення та редагування тестів

Для створення та редагування запитань створимо додаткову форму. Вид форми наведений на рисунку 3.7.

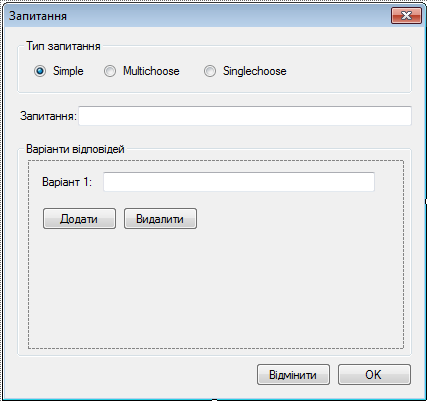


Рисунок 3.7 - Вид форми створення та редагування запитань

Для аналізу даних створимо форму на якій будуть відображатися налаштування типу відображення даних та область для відображення графічної або текстової інформації. Вид форми наведений на рисунку 3.8.

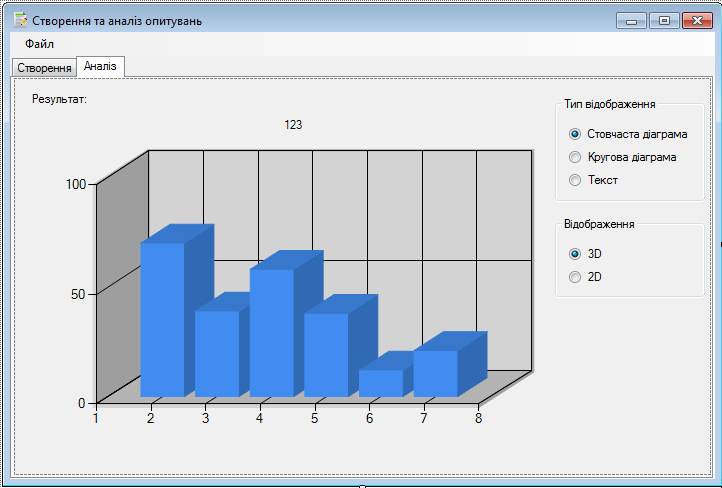


Рисунок 3.8 - Вид форми для аналізу даних

### Опис програми

При розробці програмного коду програми доцільно розбити його на окремі модулі, які відповідатимуть за окремі логічні дії та могли б легко змінюватися та масштабуватися.

Так як програма працюватиме з базою даних потрібно створити модуль для роботи з базою, написати запити та зробити інтерфейс для користування даним модулем.

Особливістю при роботі з базою є те, що підтримання зв’язку з базою є ресурсоємкістною процедурою, тому перед запросом до бази необхідно відкрити з’єднання, зробити необхідні дії і закрити базу. Щоб уникнути багатьох одночасних не закритих з’єднань створимо клас таким чином, щоб екземпляр класу був тільки одним.

Розробимо модуль для роботи з базою даних на мобільному пристрої. Модуль буде написаний на мові програмування Java та оформлений в вигляді класу.

Розглянемо детальніше важливі методи цього класу.

Метод створення таблиць:

public void onCreate(SQLiteDatabase db) {

String createTest = "CREATE TABLE " + TABLE\_TEST + " (" +

TEST\_COLUMN\_ID + " INTEGER PRIMARY KEY, " +

TEST\_COLUMN\_TITLE + " TEXT, " +

TEST\_COLUMN\_DATE + " DATETIME); ";

db.execSQL(createTest);

String createQuestion = "CREATE TABLE " +

TABLE\_QUESTION + " (" +

QUESTION\_COLUMN\_ID + " INTEGER PRIMARY KEY, " +

QUESTION\_COLUMN\_ID\_TEST + " INTEGER, " +

QUESTION\_COLUMN\_TITLE + " TEXT, " +

QUESTION\_COLUMN\_TYPE + " TEXT, " +

QUESTION\_COLUMN\_VARIANTS + " TEXT); ";

db.execSQL(createQuestion);

String createAnswer = "CREATE TABLE " +

TABLE\_ANSWER + " (" +

ANSWER\_COLUMN\_ID + " INTEGER PRIMARY KEY

AUTOINCREMENT,

" +

ANSWER\_COLUMN\_ID\_QUESTION + " INTEGER, " +

ANSWER\_COLUMN\_TEXT + " TEXT); ";

db.execSQL(createAnswer);

}

Метод видалення таблиць:

public void onUpgrade(SQLiteDatabase db, int oldVersion, int

newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_TEST);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_QUESTION);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_ANSWER);

onCreate(db);

}

Метод вибірки опитувань:

public Test[] getTests() {

ArrayList<Test> list = new ArrayList<Test>();

Cursor cursor = mDatabase.rawQuery("SELECT \* FROM " +

TABLE\_TEST, null);

cursor.moveToFirst();

if (!cursor.isAfterLast()) {

do {

int id = cursor.getInt(0);

String title = cursor.getString(1);

String date = cursor.getString(2);

date = date.replace('T', ' ');

date = date.substring(0, date.indexOf('+'));

list.add(new Test(id, title, date));

} while (cursor.moveToNext());

}

cursor.close();

Test[] array = new Test[list.size()];

for (int i = 0; i < array.length; i++) {

array[i] = list.get(i);

}

return array;

}

Метод додавання опитувань:

public long addTest(String id, String title, String createDate){

ContentValues values = new ContentValues();

values.put(TEST\_COLUMN\_ID, id);

values.put(TEST\_COLUMN\_TITLE, title);

values.put(TEST\_COLUMN\_DATE, createDate);

return mDatabase.insert(TABLE\_TEST, null, values);

}

Метод додавання запитань:

public long addQuestion(String id, String id\_test, String title, String type, String variants) {

ContentValues values = new ContentValues();

values.put(QUESTION\_COLUMN\_ID, id);

values.put(QUESTION\_COLUMN\_ID\_TEST, id\_test);

values.put(QUESTION\_COLUMN\_TITLE, title);

values.put(QUESTION\_COLUMN\_TYPE, type);

values.put(QUESTION\_COLUMN\_VARIANTS, variants);

return mDatabase.insert(TABLE\_QUESTION, null, values);

}

Метод додавання відповідей:

public long addAnswer(String id, String text) {

ContentValues values = new ContentValues();

values.put(ANSWER\_COLUMN\_ID\_QUESTION, id);

values.put(ANSWER\_COLUMN\_TEXT, text);

return mDatabase.insert(TABLE\_ANSWER, null, values);

}

### Посібник користувача

Для створення опитування необхідно створити файл XML. Пізніше буде розроблено допоміжну програму для більш зручного створення файлу XML. Тепер необхідно скопіювати цей файл на карту пам’яті мобільного пристрою. Після чого запустити відповідний додаток на мобільному пристрої. Головне вікно додатка представлено на рисунку 3.9.

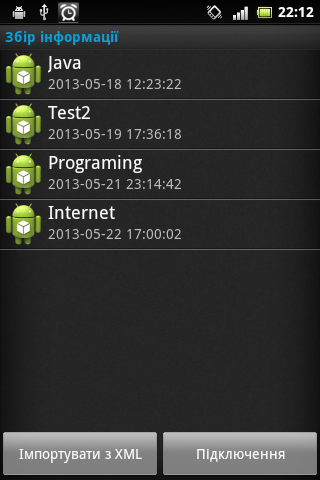


Рисунок 3.9 – Список опитувань на мобільному додатку

Для імпорту файлу опитування необхідно натиснути кнопку «Імпортувати з XML» та вибрати необхідний файл для імпорту. Після імпорту опитування відобразиться в списку опитувань. Процедура добавлення представлена на рисунку 3.10.

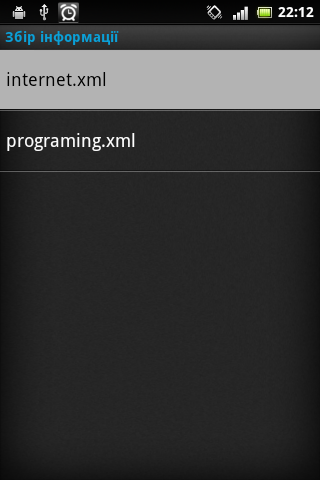


Рисунок 3.10 – Імпорт опитування з файлу

Для проходження опитування необхідно вибрати необхідне опитування. З’явиться відображення можливих дій с опитуваннями, яке представлене на рисунку 3.11.

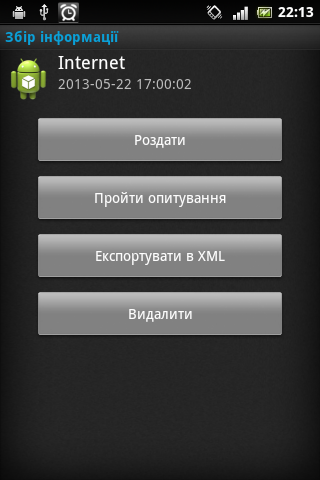


Рисунок 3.11 – Вікно з діями над опитуванням

Дія «Роздати» роздає тест по Bluetooth на віддалений пристрій. Дія «Пройти опитування» відповідає за проходження опитування на локальному пристрої. Дія «Експортувати в XML» дозволяє експортувати відповіді на запитання. Дія «Видалити» видаляє опитування з базі даних. Для проходження тесту необхідно відповісти на питання і натиснути «Завершити опитування». Вікно з проходженням опитування представлене на рисунку 3.12.

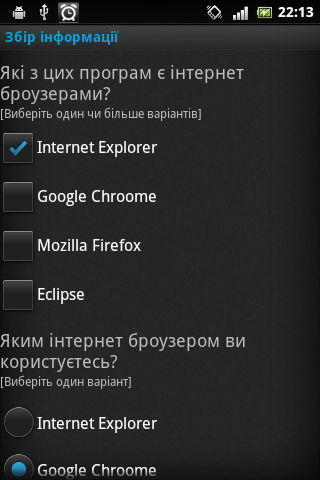


Рисунок 3.12 – Вікно з діями над опитуванням

Після проходження опитування можна експортувати відповіді в файл. При успішному експорті буде показане відповідне повідомлення. Вікно з успішним експортом представлене на рисунку 3.13.

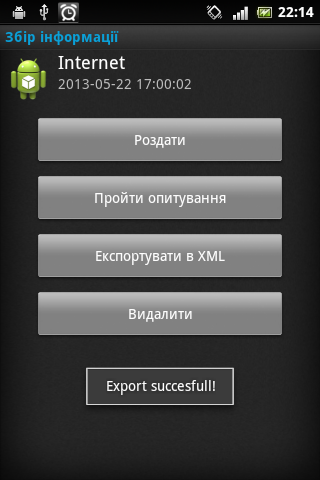


Рисунок 3.13 – Вікно з успішним експортом

Опишемо посібник користувача для допоміжної програми.

Для початку необхідно створити опитування. Для цього потрібно запустити десктопну версію програми. Для цього необхідно запустити файл «TestTool.exe». Головне вікно представлено на рисунку 3.14.

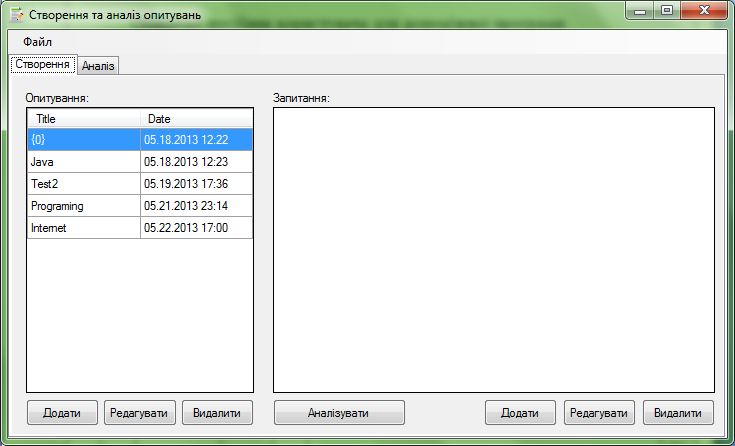


Рисунок 3.14 – Головне вікно програми

Для створення опитування необхідно натиснути на кнопку «Додати» нижче таблиці зі списком опитувань. В діалоговому вікні необхідно ввести ім’я опитування. Та натиснути кнопку «OK» Вікно з введенням імені представлено на рисунку 3.15.

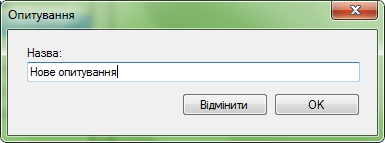


Рисунок 3.15 – Діалогове вікно введення імені

Створене опитування з’явиться в списку опитувань. Для додавання запитань необхідно під таблицею зі списком запитань натиснути кнопку «Додати». Після чого з’явиться діалогове вікно створення запитання, яке представлене на рисунку 3.16.

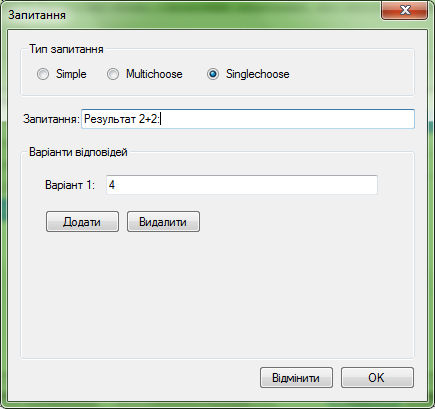


Рисунок 3.16 – Діалогове вікно створення запитання

Для створення запитання необхідно вибрати тип запитання. Типів є 3: просте запитання, яке вимагає вільної відповіді від користувача, запитання з можливістю вибору багатьох варіантів, запитання з можливістю вибору одного запитання. Створені запитання додаються до відповідного опитування. Вікно з 3-ма запитаннями до створеного опитування представлене на рисунку 3.17.

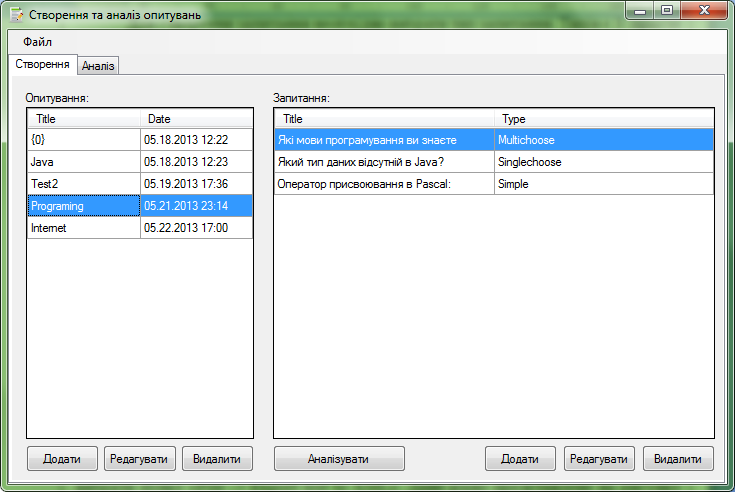


Рисунок 3.17 – Вікно з створеними запитаннями

Для експорту опитування необхідно вибрати опитування зі списку і в меню вибрати пункт «Файл -> Експортувати в XML». Дане вікно представлене на рисунку 3.18.

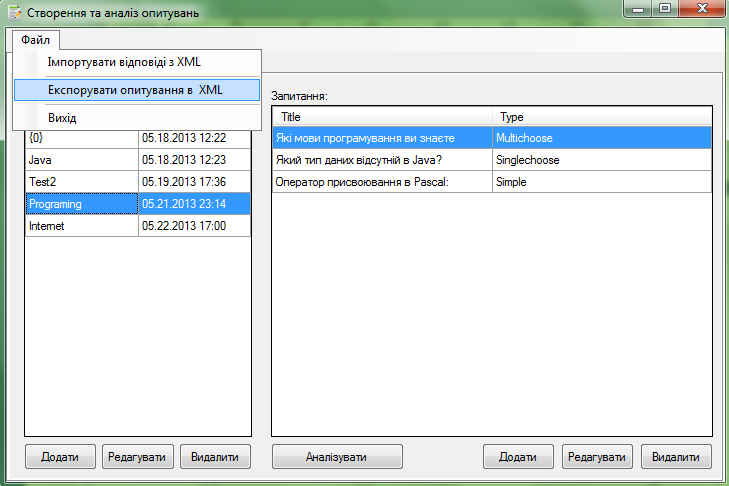


Рисунок 3.18 – Експорт опитування в файл

При експортуванні опитувань в один файл можна експортувати лише одне опитування для полегшення подальшого розбору файлу на мобільному пристрої.

Після імпорту відповідей в десктопну програму, можна вибрати запитання і натиснути «Аналізувати», після чого відобразиться вікно з графічним відображенням даних. Вікно зображене на рисунку 3.19.

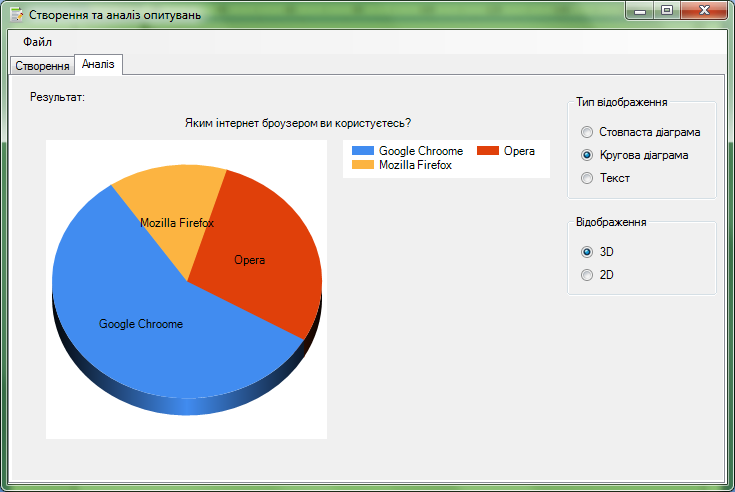


Рисунок 3.19 – Вікно аналізу запитання

Вікно аналізу запитання містить налаштування відображення. Користувач може задати тип відображення з трьох доступних, а саме в вигляді стовпасчтої діаграми, в вигляді кругової діаграми та а вигляді тексту. Для відображення типу діаграм також доступні налаштування об’ємності.

# ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

## Охорона праці

Питання охорони праці людини необхідно вирішувати на всіх стадіях трудового процесу незалежно від виду професійної діяльності. Забезпечення безпечних і здорових умов праці в значній мірі залежить від правильної оцінки небезпечних, шкідливих виробничих факторів. Однакові по складності зміни в організмі людини можуть бути викликані різними причинами. Це можуть бути фактори виробничого середовища, надмірне фізичне і розумове навантаження, нервово-емоційна напруга, а також різне сполучення цих причин.

Організація робочого місця.

Приміщення, в якому працює програміст, має загальну площу 20 м2, висоту стелі 3 м. У приміщенні знаходиться 7 робочих місць з ПК. Кожне робоче місце обладнане робочім столом площею 1,2 м2, стільцем та персональним комп'ютером, що складається з монітора, системного блоку, клавіатури та миші. Слід відзначити, що площа одного робочого місця оператора ПК не повинна бути меншою за 6м2, а об'єм не менший за 20м3, тобто площі та об'єму даного приміщення не вистачає для розташування 7 робочих місць операторів ПК. Аналіз умов праці показує, що у приміщенні лабораторії на програміста можуть негативно впливати наступні фізичні та психофізіологічні фактори:

* підвищена або знижена температура повітря робочої зони;
* підвищена або знижена вологість повітря;
* недостатня освітленість робочого місця;
* підвищений рівень шуму на робочому місці;
* підвищена іонізація повітря;
* підвищений рівень електромагнітних випромінювань;
* нервово-психічні перевантаження (розумова перенапруга, перенапруга аналізаторів);
* фізичні перевантаження (одноманітна поза викликає статичну втому).

Мікроклімат робочої зони програміста.

Робота програміста за енерговитратами відноситься до категорії легких робіт, тому повинні дотримуватися наступні вимоги згідно ДСН 3.3.6.042- 99 – оптимальна температура повітря – 220С (допустима – 20-240С), оптимальна відносна вологість – 40-60% (допустима – не більш 75%), швидкість руху повітря не більш 0,1 м/с. Виміряні за допомогою приладів (психрометр Августа) температура та вологість у лабораторії відповідають вказаним у таблиці для теплого періоду року. Розташовані у приміщенні 7 ПК являються джерелами тепловиділень, крім того для підтримання у приміщенні в холодний період року оптимальних параметрів мікроклімату використовуються нагріті поверхні опалювальної системи. Нормованим показником ІЧВ являється гранично допустима густина потоку енергії Іг.д, Вт/м2, яка встановлюється в залежності від площі опромінюваної поверхні тіла людини (Sопр). Нормовані рівні складають: Іг.д =35 Вт/м2 при Sопр > 50%; Іг.д =70 Вт/м2 при Sопр ~ 25-50%; Іг.д =100 Вт/м2 при Sопр < 25%.

Правильно спроектоване і виконане виробниче освітлення покращує умови зорової роботи, знижує стомлюваність, сприяє підвищенню продуктивності праці, благотворно впливає на виробниче середовище, надаючи позитивну психологічну дію на працюючого, підвищує безпеку праці і знижує травматизм.

Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань, тому такий важливий правильний розрахунок освітленості.

Існує три види освітлення - природне, штучне і поєднане (природне і штучне разом).

Природне освітлення - освітлення приміщень денним світлом, що потрапляє через світлові прорізи в зовнішніх огороджуючих конструкціях приміщення. Природне освітлення характеризується тим, що змінюється в широких межах залежно від часу дня, пори року, характеру області і ряду інших чинників.

Штучне освітлення застосовується при роботі в темний час доби і вдень, коли не вдається забезпечити нормовані значення коефіцієнта природного освітлення (похмура погода, короткий світловий день). Освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним, називається змішаним освітленням.

Штучне освітлення підрозділяється на робоче, аварійне, евакуаційне, охоронне. Робоче освітлення, у свою чергу, може бути загальним або комбінованим. Загальне - освітлення, при якому світильники розміщуються у верхній зоні приміщення рівномірно, або, як розташоване устаткування. Комбіноване - освітлення, при якому до загального додається місцеве освітлення.

При виконанні робіт категорії високої зорової точності (найменший розмір об'єкту розрізнення 0,3 ... 0,5 мм) величина коефіцієнта природного освітлення (КЕО) повинна бути не нижче 1,5%, а при зоровій роботі середньої точності (найменший розмір об'єкту розрізнення 0,5 ... 1,0 мм) КЕО повинен бути не нижче 1,0%. В якості джерел штучного освітлення звичайно використовуються люмінесцентні лампи типа ЛБ, або ДРЛ, які попарно об'єднуються в світильники, які повинні розташовуватися рівномірно над робочими поверхнями.

Вимоги до освітленості в приміщеннях, де встановлені комп'ютери, наступні: при виконанні зорових робіт високої точності загальна освітленість повинна складати 300лк, а комбінована - 750лк; аналогічні вимоги при виконанні робіт середньої точності - 200 і 300лк відповідно.

Крім того все поле зору повинне бути освітлено достатньо рівномірно - ця основна гігієнічна вимога. Іншими словами, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими, оскільки яскраве світло в районі периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності.

Параметри мікроклімату .

Параметри мікроклімату можуть мінятися в широких межах, у той час як необхідною умовою життєдіяльності людини є підтримка постійності температури тіла завдяки терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище. Принцип нормування мікроклімату - створення оптимальних умов для теплообміну тіла людини з навколишнім середовищем.

Обчислювальна техніка є джерелом істотних тепловиділень, що може привести до підвищення температури і зниження відносної вологості в приміщенні. У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах СН-245-71 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (таблиця 4.1) .

Об'єм приміщень, в яких розміщені працівники обчислювальних центрів, не повинен бути меншим 19,5 м 3 / людини з урахуванням максимального числа одночасно працюючих в зміну. Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери, приведені в таблиці 4.2.

Таблиця 4.1 Параметри мікроклімату для приміщень, де встановлені комп'ютери

|  |  |  |
| --- | --- | --- |
| Період року | Параметр мікроклімату | Величина |
| Холодний | Температура повітря в приміщенні  Відносна вологість  Швидкість руху повітря | 22 ... 24 ° С  40 ... 60%  до 0,1 м / с |
| Теплий | Температура повітря в приміщенні  Відносна вологість  Швидкість руху повітря | 23 ... 25 ° С  40 ... 60%  0,1 ... 0,2 м / с |

Таблиця 4.2 Норми подачі свіжого повітря в приміщення, де розташовані комп'ютери

|  |  |
| --- | --- |
| Характеристика приміщення | Об'ємна витрата подається в приміщення свіжого повітря, м3 / на одну людину в годину |
| Об'єм до 20м3 на особу  20 ... 40м3 на особу  Більш 40м3 на особу | Не менше 30  Не менше 20  Природна вентиляція |

Для забезпечення комфортних умов використовуються як організаційні методи (раціональна організація проведення робіт залежно від пори року і доби, чергування праці і відпочинку), так і технічні засоби (вентиляція, кондиціювання повітря, опалювальна система).

Шум погіршує умови праці надаючи шкідливу дію на організм людини. Працюючі в умовах тривалої шумової дії випробовують дратівливість, головні болі, запаморочення, зниження пам'яті, підвищену стомлюваність, зниження апетиту, біль у вухах і т.д. Такі порушення в роботі ряду органів і систем організму людини можуть викликати негативні зміни в емоційному стані людини аж до стресових. Під впливом шуму знижується концентрація уваги, порушуються фізіологічні функції, з'являється втома у зв'язку з підвищеними енергетичними витратами і нервово-психічним напруженням, погіршується мовна комутація. Все це знижує працездатність людини і її продуктивність, якість і безпеку праці. Тривала дія інтенсивного шуму [вище 80 дБ (А)] на слух людини приводить до його часткової або повної втрати.

У таблиці 4.3 вказані граничні рівні звуку залежно від категорії тяжкості і напруженості праці, що є безпечними відносно збереження здоров'я і працездатності.

Таблиця 4.3 - Граничні рівні звуку, дБ, на робочих місцях.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категорія  напруженості праці | Категорія важкості праці | | | |
| I. Легка | II. Середня | III. Важка | IV. Дуже важка |
| I. Мало напружений | 80 | 80 | 75 | 75 |
| II. Помірно напружений | 70 | 70 | 65 | 65 |
| III. Напружений | 60 | 60 | - | - |
| IV. Дуже напружений | 50 | 50 | - | - |

Рівень шуму на робочому місці математиків-програмістів і операторів відеоматеріалів не повинен перевищувати 50дБА, а в залах обробки інформації на обчислювальних машинах - 65дБА. Для зниження рівня шуму стіни і стеля приміщень, де встановлені комп'ютери, можуть бути облицьовані звукопоглинальними матеріалами. Рівень вібрації в приміщеннях обчислювальних центрів може бути понижений шляхом встановлення устаткування на спеціальні віброізолятори.

Більшість вчених вважають, що як короткочасне, так і тривалий вплив усіх видів випромінювання від екрану монітора не небезпечно для здоров'я персоналу, що обслуговує комп'ютери. Проте вичерпних даних щодо небезпеки дії випромінювання від моніторів на працюючих з комп'ютерами не існує і дослідження в цьому напрямі продовжуються.

Допустимі значення параметрів неіонізуючих електромагнітних випромінювань від монітора комп'ютера представлені в таблиці 4.4.

Максимальний рівень рентгенівського випромінювання на робочому місці оператора комп'ютера звичайно не перевищує 10мкбер / ч, а інтенсивність ультрафіолетового і інфрачервоного випромінювань від екрану монітора лежить в межах 10 ... 100МВт / м2.

Таблиця 4.4 - Допустимі значення параметрів неіонізуючих електромагнітних випромінювань (відповідно до СанПіН 2.2.2.542-96)

|  |  |
| --- | --- |
| Найменування параметра | Допустимі значення |
| Напруженість електричної складової електромагнітного поля на відстані 50см від поверхні відеомонітора | 10В / м |
| Напруженість магнітної складової електромагнітного поля на відстані 50см від поверхні відеомонітора | 0,3 А / м |
| Напруженість електростатичного поля не повинна перевищувати:   * для дорослих користувачів * для дітей дошкільних установ і що вчаться середніх спеціальних і вищих навчальних закладів | 20кВ / м  15кВ / м |

Для зниження дії цих видів випромінювання рекомендується застосовувати монітори із зниженим рівнем випромінювання (MPR-II, TCO-92, TCO-99), встановлювати захисні екрани, а також дотримуватися регламентовані режими праці та відпочинку.

Закон України "Про охорону праці" є основоположним законодавчим документом в галузі охорони праці. Його дія поширюється на всі підприємства, установи і організації незалежно від форм власності та видів діяльності на усіх громадян, які працюють, а також залучені до праці на цих підприємствах. Закон визначає основні положення щодо реалізації конституційного права громадян про охорону їх життя і здоров'я в процесі трудової реальності, регулює відносини між власниками підприємства, установи чи організації або уповноваженим ним органом і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні.

Основним нормативним документом щодо забезпечення охорони праці користувачів ВДТ є ДНАОП 0,00-1.31-99 "Правила охорони праці під час експлуатації електронно-обчислювальних машин". Дані Правила встановлюють вимоги безпеки та санітарно - гігієнічні вимоги до обладнання робочих місць користувачів електронно-обчислювальних машин і персональних комп'ютерів (дані ЕОМ) та працівників, що виконують обслуговування, ремонт та налагодження ЕОМ та роботи з застосуванням ЕОМ, відповідно до сучасного стану техніки та наукових досліджень у сфері безпечної організації робіт з експлуатації ЕОМ та з урахуванням положень міжнародних нормативно-правових актів з цих питань.

ДНАОП 0.00-1.31-99 поширюється на всі підприємства, установи, організації, юридичні особи, крім зазначених нижче, незалежно від форми власності і відомчої належності і видів діяльності та на фізичних осіб, що займаються підприємницькою діяльністю з правом найму робочої сили, які здійснюють розробку, виробництво та застосування ЕОМ, у тому числі й на тих, які мають робочі місця, обладнані ЕОМ і або виконують обслуговування, ремонт та налагодження ЕОМ.

## Безпека життєдіяльності

З розвитком науково-технічного прогресу важливу роль грає можливість безпечного виконання людьми своїх трудових обов'язків. У зв'язку з цим була створена і розвивається наука про безпеку праці і життєдіяльності людини.

Безпека життєдіяльності (БЖД) - це комплекс заходів, спрямованих на забезпечення безпеки людини в середовищі проживання, збереження його здоров'я, розробку методів і засобів захисту шляхом зниження впливу шкідливих і небезпечних факторів до допустимих значень, вироблення заходів по обмеженню збитку в ліквідації наслідків надзвичайних ситуацій мирного і воєнного часу.

Мета і зміст БЖД:

* виявлення і вивчення чинників навколишнього середовища, що негативно впливають на здоров'я людини;
* ослаблення дії цих чинників до безпечних меж або виключення їх якщо це можливо;
* ліквідація наслідків катастроф і стихійних лих.

Коло практичних задач БЖД перш за все обумовлений вибором принципів захисту, розробкою і раціональним використанням засобів захисту людини і природного середовища від впливу техногенних джерел і стихійних явищ, а також коштів, що забезпечують комфортний стан середовища життєдіяльності.

Охорона здоров'я трудящих, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму складає одну з головних турбот людського суспільства. Звертається увага на необхідність широкого застосування прогресивних форм наукової організації праці, зведення до мінімуму ручної, малокваліфікованої праці, створення обстановки, що виключає професійні захворювання і виробничий травматизм.

На робочому місці повинні бути передбачені заходи захисту від можливого впливу небезпечних і шкідливих факторів виробництва. Рівні цих чинників не повинні перевищувати граничних значень, обумовлених правовими, технічними та санітарно-технічними нормами. Ці нормативні документи зобов'язують до створення на робочому місці умов праці, при яких вплив небезпечних і шкідливих чинників на працюючих або усунуто зовсім, або знаходиться в допустимих межах.

Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової праці. Їх праця стала більш інтенсивним, напруженим, які вимагають значних витрат розумової, емоційної і фізичної енергії. Це зажадало комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку.

В даний час комп'ютерна техніка широко застосовується у всіх областях діяльності людини. При роботі з комп'ютером людина піддається дії ряду небезпечних і шкідливих виробничих факторів: електромагнітних полів (діапазон радіочастот: ВЧ, УВЧ і СВЧ), інфрачервоного і іонізуючого випромінювань, шуму і вібрації, статичної електрики і ін.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ. Велике значення має раціональна конструкція і розташування елементів робочого місця, що важливо для підтримки оптимальної робочої пози людини-оператора.

У процесі роботи з комп'ютером необхідно дотримувати правильний режим праці та відпочинку. В іншому випадку у персоналу наголошуються значна напруга зорового апарату з появою скарг на незадоволеність роботою, головні болі, дратівливість, порушення сну, втому і хворобливі відчуття в очах, в поясниці, в області шиї і руках.

Проектування робочих місць, забезпечених відеотерміналами, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки.

Робоче місце і взаємне розташування всіх його елементів повинне відповідати антропометричним, фізичним і психологічним вимогам. Велике значення має також характер роботи. Зокрема, при організації робочого місця програміста повинні бути дотримані наступні основні умови: оптимальне розміщення устаткування, що до складу робочого місця і достатній робочий простір, що дозволяє здійснювати всі необхідні рухи і переміщення.

Ергономічними аспектами проектування відеотермінальних робочих місць, зокрема, є: висота робочої поверхні, розміри простору для ніг, вимоги до розташування документів на робочому місці (наявність і розміри підставки для документів, можливість різного розміщення документів, відстань від очей користувача до екрану, документа, клавіатури і т.д.), характеристики робочого крісла, вимоги до поверхні робочого столу, регульованість елементів робочого місця.

Головними елементами робочого місця програміста є стіл і крісло. Основним робочим положенням є положення сидячи.

Робоча поза сидячи викликає мінімальне стомлення програміста. Раціональне планування робочого місця передбачає чіткий порядок і сталість розміщення предметів, засобів праці і документації. Те, що потрібно для виконання робіт частіше, розташоване в зоні легкої досяжності робочого простору.

Моторне поле - простір робочого місця, в якому можуть здійснюватися рухові дії людини.

Максимальна зона досяжності рук - це частина моторного поля робочого місця, обмеженого дугами, описуваними максимально витягнутими руками при русі їх у плечовому суглобі.

Оптимальна зона - частина моторного поля робочого місця, обмеженого дугами, описуваними передпліччями при русі в ліктьових суглобах з опорою в точці ліктя і з відносно нерухомим плечем.

Для комфортної роботи стіл повинен задовольняти наступним умовам:

* висота столу повинна бути вибрана з урахуванням можливості сидіти вільно, в зручній позі, при необхідності спираючись на підлокітники;
* нижня частина столу повинна бути сконструйована так, щоб програміст міг зручно сидіти, не був змушений підбирати ноги;
* поверхня столу повинна мати властивості, що виключають появу відблисків у поле зору програміста;
* конструкція столу повинна передбачати наявність висувних ящиків (не менше 3 для зберігання документації, лістингів, канцелярських приналежностей).
* висота робочої поверхні рекомендується в межах 680-760мм. Висота поверхні, на яку встановлюється клавіатура, повинна бути близько 650мм.

Велике значення надається характеристикам робочого крісла. Так, рекомендована висота сидіння над рівнем підлоги перебуває в межах 420-550мм. Поверхня сидіння м'яка, передній край закруглений, а кут нахилу спинки - регульований.

Необхідно передбачати при проектуванні можливість різного розміщення документів: збоку від відеотерміналу, між монітором і клавіатурою і т.п. Крім того, у випадках, коли відеотермінал має низьку якість зображення, наприклад помітні мелькання, відстань від очей до екрану роблять більше (біля 700мм), ніж відстань від ока до документа (300-450мм). Взагалі при високій якості зображення на відеотерміналі відстань від очей користувача до екрану, документа і клавіатури може бути рівним.

Положення екрану визначається:

* відстанню зчитування (0,6 ... 0,7 м);
* кутом зчитування, напрямком погляду на 20° нижче горизонталі до центру екрану, причому екран перпендикулярний цьому напрямку.

Повинна також передбачатися можливість регулювання екрану:

* по висоті +3 см;
* по нахилу від -10° до +20° щодо вертикалі;
* в лівому і правому напрямках.

Велике значення також надається правильній робочій позі користувача. При незручній робочій позі можуть з'явитися болі в м'язах, суглобах і сухожиллях. Вимоги до робочої пози користувача відеотермінала наступні:

* голова не повинна бути нахилена більш ніж на 20°,
* плечі повинні бути розслаблені,
* лікті - під кутом 80° ... 100°,
* передпліччя і кисті рук - в горизонтальному положенні.

Причина неправильної пози користувачів обумовлена ​​наступними чинниками: немає хорошої підставки для документів, клавіатура знаходиться дуже високо, а документи - низько, нікуди покласти руки і кисті, недостатній простір для ніг.

З метою подолання вказаних недоліків даються загальні рекомендації: краще пересувна клавіатура; повинні бути передбачені спеціальні пристосування для регулювання висоти столу, клавіатури і екрану, а також підставка для рук.

Істотне значення для продуктивної і якісної роботи на комп'ютері мають розміри знаків, густину їх розміщення, контраст і співвідношення яскравості символів і фону екрану. Якщо відстань від очей оператора до екрану дисплея складає 60 ... 80 см, то висота знака повинна бути не менше 3мм, оптимальне співвідношення ширини і висоти знака складає 3:4, а відстань між знаками - 15 ... 20% їх висоти. Співвідношення яскравості фону екрану і символів - від 1:2 до 1:15.

Під час користування комп'ютером медики радять встановлювати монітор на відстані 50-60 см від очей. Фахівці також вважають, що верхня частина відеодисплея повинна бути на рівні очей або трохи нижче. Коли людина дивиться прямо перед собою, його очі відкриваються ширше, ніж коли він дивиться вниз. За рахунок цього площа огляду значно збільшується, викликаючи обезводнення очей. До того ж якщо екран встановлений високо, а очі широко відкриті, порушується функція моргання. Це значить, що очі не закриваються повністю, не омиваються слізною рідиною, не одержують достатнього зволоження, що приводить до їх швидкої стомлюваності.

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці.

# ВИСНОВКИ

У даній роботі було проведено дослідження необхідності розробки програмної системи для сбору інформації за допомогою мобільних пристроїв.

На етапі проектування системи була розроблена функціональна схема роботи, яка демонструє взаємодію програми для Windows та програми для Android. Також були розроблені та описані алгоритми роботи підсистем кожної із програм.

В результаті проведених дій було розроблено програмне забезпечення, яке дозволяє проводити опитування групи людей за допомогою мобільний пристроїв. Була розроблена програма для операційної системи android, яка на разі є широкорозповсюдженою як на смартфонах так і на планшетних комп’ютерах.

При створення такої системи були виконані наступні завдання:

* розробка структури програмної системи, її алгоритмів роботи;
* вибір системи керування базами даних та розробка структури бази даних;
* створення механізму взаємодії частин програмної системи;
* вибір інструментів для створення програмних засобів;
* створення опитувань з запитаннями різного типу: просте запитання, де користувач сам пише відповідь, запитання з одним варіантом вибору, запитання з багатьма варіантами вибору;
* розроблений модуль аналізу отриманих даних від користувачів.

Розроблена програма дозволить пришвидшити збір інформації від користувачів так як використовує мобільні пристрої, які наразі широко розповсюдженні і можуть використовуватися для збору інформації.

Для спрощення використання мобільної версії програми була розроблена обслуговуюча програма для персонального комп’ютера, що дозволяє зручно створювати опитування та аналізувати відповіді. Перевагами обслуговуючої програми є наявність повноцінної клавіатури, великого монітору персональних комп’ютерів порівняно з мобільними пристроями.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Джеффри Рихтер. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. – П.:Русская Редакция, 2008. – 656 с.
2. Герберт Шилдт. C# 4.0. Полное руководство. - Вильямс, 2013, – 1056 стр.
3. Майк Гандэрлой. ADO и ADO.NET. Полное руководство. - Век +, 2011, – 908 стр.
4. Айан Брейс. Анкетирование. Разработка опросных листов, их роль и значение при проведении рыночных исследований. - Баланс Бизнес Букс, 2005, – 336 стр.
5. Татарова Г.Г. Методология анализа данных в социологии. - М.: NOTA BENE, 1999, – 224 стр.
6. Брюс Эккель. Философия Java. – П., 2004. – 1105 с.
7. Ильдар Хабибуллин. Java 7. – БХВ-Петербург, 2012. – 768 с.
8. В. Монахов. Язык программирования Java и среда NetBeans. – БХВ-Петербург, 2011. – 704 с.
9. Роберт Седжвик, Кевин Уэйн. Алгоритмы на Java. – Вильямс, 2012. – 848 с.
10. Сильвен Ретабоуил. Android NDK. Разработка приложений под Android на С/С++. – ДМК Пресс, 2012. – 496 с.
11. Денис Колисниченко. Android для пользователя. Полезные программы и советы. – БХВ-Петербург, 2013. – 256 с.
12. С. Хашими, С. Коматинени, Д. Маклинг. Разработка приложений для Android. – Москва, 2001. – 987 с.
13. Крёнке Д. Теория и практика построения баз данных, 8-е изд. - П.:"Питер", 2003, – 800 стр.
14. М. П. Малыхина. Базы данных. Основы, проектирование, использование. - БХВ-Петербург, 2006, – 528 стр.
15. В. В. Дунаев. Базы данных. Язык SQL для студента. - П.:"Питер", 2006, – 288 стр.
16. К. Дж. Дейт. SQL и реляционная теория. Как грамотно писать код на SQL. - Символ-Плюс, 2010, – 480 стр.
17. Інтернет сайти:

* Android Developers

<http://developer.android.com/index.html>

* MSDN Library

<http://msdn.microsoft.com/en-us/library/ms123401.aspx>

* Habrahabr

<http://habrahabr.ru/post/133575/>

* Lars Vogel Blog

<http://www.vogella.com/articles/AndroidListView/article.html>

* StackOverflow

[http://stackoverflow.com/](http://stackoverflow.com/questions/7741804/android-open-file)

* Datacol

http://web-data-extractor.net/

* IPO Statistics

<http://cti.lnc.ru/cti/ips/>

* Tec-IT

<http://www.tec-it.com/ru/software/data-acquisition/twedgece/pocket-pc/Default.aspx>

# ДОДАТОК А

MainActivity.java:

package com.andrey.pzas091.statisticscollection;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.util.UUID;

import org.xmlpull.v1.XmlPullParser;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.bluetooth.BluetoothSocket;

import android.content.Intent;

import android.os.Bundle;

import android.util.Xml;

import android.view.View;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ListView;

import android.widget.Toast;

public class MainActivity extends Activity {

private final int OPEN\_FILE = 12;

private ListView lvTest;

private BluetoothAdapter mBluetoothAdapter;

private final int REQUEST\_ENABLE\_BT = 16;

private final int BT\_DEVICE = 17;

private final int ANSWER\_REQUEST = 18;

public static BluetoothDevice device;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

lvTest = (ListView)findViewById(R.id.lvTest);

lvTest.setOnItemClickListener(new OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> a, View parent, int pos, long id) {

Test item = (Test) lvTest.getAdapter().getItem(pos);

TestActivity.item = item;

Intent intent = new Intent(lvTest.getContext(), TestActivity.class);

startActivity(intent);

}

});

showTestList();

}

private void showTestList() {

DataBase db = new DataBase(this);

Test[] data = db.getTests();

db.close();

if (data != null) {

TestArrayAdapter adapter = new TestArrayAdapter(this, data);

lvTest.setAdapter(adapter);

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (requestCode == OPEN\_FILE && resultCode == RESULT\_OK) {

if (data.hasExtra("path")) {

String fileName = data.getStringExtra("path");

try {

parseXml(fileName);

showTestList();

}

catch (Exception e) {

e.printStackTrace();

}

}

}

if (requestCode == REQUEST\_ENABLE\_BT && resultCode == RESULT\_OK) {

startClient();

}

if (requestCode == BT\_DEVICE && resultCode == RESULT\_OK) {

getQuestion();

}

if (requestCode == ANSWER\_REQUEST && resultCode == RESULT\_OK) {

if (data.hasExtra("result")) {

String result = data.getStringExtra("result");

try {

mmSocket.getOutputStream().write(result.getBytes());

mmSocket.close();

}

catch (IOException e) {

e.printStackTrace();

return;

}

}

}

super.onActivityResult(requestCode, resultCode, data);

}

private void parseXml(String fileName) throws Exception {

XmlPullParser parser = Xml.newPullParser();

parser.setFeature(XmlPullParser.FEATURE\_PROCESS\_NAMESPACES, false);

InputStream in = new FileInputStream(fileName);

parser.setInput(in, null);

boolean first = true;

//Test Data

String testId = null;

String testTitle = null;

String testCreateDate = null;

//Questions Data

String questionId = null;

String questionTestId = null;

String questionTitle = null;

String questionType = null;

String questionVariants = null;

//Parse XML

while (parser.next() != XmlPullParser.END\_DOCUMENT) {

if (parser.getEventType() != XmlPullParser.START\_TAG) {

continue;

}

String name = parser.getName();

if (name.equals("Table")) {

DataBase db = new DataBase(this);

try {

if (first) {

db.addTest(testId, testTitle, testCreateDate);

first = false;

}

}

catch (Exception ex) {

}

db.addQuestion(questionId, questionTestId, questionTitle,

questionType, questionVariants);

db.close();

}

if (parser.next() == XmlPullParser.TEXT) {

String result = parser.getText();

if (name.equals("test.\_id")) {

testId = result;

}

if (name.equals("test.title")) {

testTitle = result;

}

if (name.equals("create\_date")) {

testCreateDate = result;

}

if (name.equals("question.\_id")) {

questionId = result;

}

if (name.equals("id\_test")) {

questionTestId = result;

}

if (name.equals("question.title")) {

questionTitle = result;

}

if (name.equals("type")) {

questionType = result;

}

if (name.equals("variants")) {

questionVariants = result;

}

parser.nextTag();

}

}

DataBase db = new DataBase(this);

db.addQuestion(questionId, questionTestId, questionTitle,

questionType, questionVariants);

db.close();

in.close();

}

public void bImport\_Click(View v) {

Intent intent = new Intent(this, OpenFileActivity.class);

startActivityForResult(intent, OPEN\_FILE);

}

public void bConnect\_Click(View view) {

mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

if (mBluetoothAdapter == null) {

Toast.makeText(getApplicationContext(), "Не знайдено Bluetooth!",

Toast.LENGTH\_SHORT).show();

return;

}

if (!mBluetoothAdapter.isEnabled()) {

Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION\_REQUEST\_ENABLE);

startActivityForResult(enableBtIntent, REQUEST\_ENABLE\_BT);

}

else {

startClient();

}

}

private void startClient() {

BluetoothPoints.mBluetoothAdapter = mBluetoothAdapter;

Intent intent = new Intent(this, BluetoothPoints.class);

startActivityForResult(intent, BT\_DEVICE);

}

private BluetoothSocket mmSocket;

private void getQuestion() {

try {

mmSocket = device.createRfcommSocketToServiceRecord(new UUID(1, 200));

mBluetoothAdapter.cancelDiscovery();

mBluetoothAdapter.cancelDiscovery();

try {

mmSocket.connect();

mmSocket.getOutputStream().write("request".getBytes());

InputStream is = mmSocket.getInputStream();

if (is.available() > 0) {

byte[] buffer = new byte[is.available()];

is.read(buffer);

AnswerActivity.questions = new String(buffer);

Intent intent = new Intent(this, AnswerActivity.class);

startActivityForResult(intent, ANSWER\_REQUEST);

}

}

catch (IOException connectException) {

try {

mmSocket.close();

} catch (IOException closeException) { }

return;

}

}

catch (IOException e) {

e.printStackTrace();

}

}

}

DataBase.java:

package com.andrey.pzas091.statisticscollection;

import java.util.ArrayList;

import android.content.ContentValues;

import android.content.Context;

import android.database.Cursor;

import android.database.sqlite.SQLiteDatabase;

import android.database.sqlite.SQLiteOpenHelper;

public class DataBase {

private static final String DATABASE\_NAME = "testing.db";

private static final String TABLE\_TEST = "test";

private static final String TEST\_COLUMN\_ID = "\_id";

private static final String TEST\_COLUMN\_TITLE = "title";

private static final String TEST\_COLUMN\_DATE = "create\_date";

private static final String TABLE\_QUESTION = "question";

private static final String QUESTION\_COLUMN\_ID = "\_id";

private static final String QUESTION\_COLUMN\_ID\_TEST = "id\_test";

private static final String QUESTION\_COLUMN\_TITLE = "title";

private static final String QUESTION\_COLUMN\_TYPE = "type";

private static final String QUESTION\_COLUMN\_VARIANTS = "variants";

private static final String TABLE\_ANSWER = "answer";

private static final String ANSWER\_COLUMN\_ID = "\_id";

private static final String ANSWER\_COLUMN\_ID\_QUESTION = "id\_question";

private static final String ANSWER\_COLUMN\_TEXT = "answer\_text";

private static final int DATABASE\_VERSION = 1;

private SQLiteDatabase mDatabase = null;

public DataBase(Context context) {

OpenHelper openHelper = new OpenHelper(context);

mDatabase = openHelper.getWritableDatabase();

}

public Test[] getTests() {

ArrayList<Test> list = new ArrayList<Test>();

Cursor cursor = mDatabase.rawQuery("SELECT \* FROM " + TABLE\_TEST, null);

cursor.moveToFirst();

if (!cursor.isAfterLast()) {

do {

int id = cursor.getInt(0);

String title = cursor.getString(1);

String date = cursor.getString(2);

date = date.replace('T', ' ');

date = date.substring(0, date.indexOf('+'));

list.add(new Test(id, title, date));

} while (cursor.moveToNext());

}

cursor.close();

Test[] array = new Test[list.size()];

for (int i = 0; i < array.length; i++) {

array[i] = list.get(i);

}

return array;

}

public String getQuestions(String id\_test) {

String result = "";

Cursor cursor = mDatabase.query(TABLE\_QUESTION, new String[] {

QUESTION\_COLUMN\_ID,

QUESTION\_COLUMN\_TITLE,

QUESTION\_COLUMN\_TYPE,

QUESTION\_COLUMN\_VARIANTS},

"id\_test = ?",

new String[] {id\_test}, null, null, null);

cursor.moveToFirst();

if (!cursor.isAfterLast()) {

do {

int id = cursor.getInt(0);

String title = cursor.getString(1);

String type = cursor.getString(2);

String variants = cursor.getString(3);

result += id + "`" + title + "`" + type + "`" + variants + "a1b2c3";

} while (cursor.moveToNext());

}

cursor.close();

return result;

}

public long addAnswer(String id, String text) {

ContentValues values = new ContentValues();

values.put(ANSWER\_COLUMN\_ID\_QUESTION, id);

values.put(ANSWER\_COLUMN\_TEXT, text);

return mDatabase.insert(TABLE\_ANSWER, null, values);

}

public long addTest(String id, String title, String createDate) {

ContentValues values = new ContentValues();

values.put(TEST\_COLUMN\_ID, id);

values.put(TEST\_COLUMN\_TITLE, title);

values.put(TEST\_COLUMN\_DATE, createDate);

return mDatabase.insert(TABLE\_TEST, null, values);

}

public long addQuestion(String id, String id\_test, String title, String type, String variants) {

ContentValues values = new ContentValues();

values.put(QUESTION\_COLUMN\_ID, id);

values.put(QUESTION\_COLUMN\_ID\_TEST, id\_test);

values.put(QUESTION\_COLUMN\_TITLE, title);

values.put(QUESTION\_COLUMN\_TYPE, type);

values.put(QUESTION\_COLUMN\_VARIANTS, variants);

return mDatabase.insert(TABLE\_QUESTION, null, values);

}

public void exportData(Exporter exp, int testId) {

Cursor cursor = mDatabase.rawQuery("SELECT \* FROM " + TABLE\_ANSWER + " WHERE id\_question IN (SELECT \_id FROM question WHERE question.id\_test = " + testId + ")", null);

cursor.moveToFirst();

if (!cursor.isAfterLast()) {

do {

int id = cursor.getInt(0);

int id\_question = cursor.getInt(1);

String text = cursor.getString(2);

exp.addRow("" + id, "" + id\_question, text);

} while (cursor.moveToNext());

}

cursor.close();

exp.close();

}

public void close() {

mDatabase.close();

}

private class OpenHelper extends SQLiteOpenHelper {

public OpenHelper(Context context) {

super(context, DATABASE\_NAME, null, DATABASE\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase db) {

String createTest = "CREATE TABLE " + TABLE\_TEST + " (" +

TEST\_COLUMN\_ID + " INTEGER PRIMARY KEY, " +

TEST\_COLUMN\_TITLE + " TEXT, " +

TEST\_COLUMN\_DATE + " DATETIME); ";

db.execSQL(createTest);

String createQuestion = "CREATE TABLE " + TABLE\_QUESTION + " (" +

QUESTION\_COLUMN\_ID + " INTEGER PRIMARY KEY, " +

QUESTION\_COLUMN\_ID\_TEST + " INTEGER, " +

QUESTION\_COLUMN\_TITLE + " TEXT, " +

QUESTION\_COLUMN\_TYPE + " TEXT, " +

QUESTION\_COLUMN\_VARIANTS + " TEXT); ";

db.execSQL(createQuestion);

String createAnswer = "CREATE TABLE " + TABLE\_ANSWER + " (" +

ANSWER\_COLUMN\_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +

ANSWER\_COLUMN\_ID\_QUESTION + " INTEGER, " +

ANSWER\_COLUMN\_TEXT + " TEXT); ";

db.execSQL(createAnswer);

}

@Override

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_TEST);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_QUESTION);

db.execSQL("DROP TABLE IF EXISTS " + TABLE\_ANSWER);

onCreate(db);

}

}

}

AnswerActivity.java:

package com.andrey.pzas091.statisticscollection;

import java.util.HashMap;

import java.util.Iterator;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.view.View.OnClickListener;

import android.widget.Button;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.LinearLayout;

import android.widget.LinearLayout.LayoutParams;

import android.widget.RadioButton;

import android.widget.RadioGroup;

import android.widget.ScrollView;

import android.widget.TextView;

public class AnswerActivity extends Activity {

public static String questions;

private LinearLayout root;

private LinearLayout.LayoutParams lp;

private LinearLayout.LayoutParams hintLp;

private LinearLayout.LayoutParams lcheckBox;

private LinearLayout.LayoutParams lEdit;

private HashMap<String, String> hash;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

hash = new HashMap<String, String>();

lp = new LinearLayout.LayoutParams

(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

lcheckBox = new LinearLayout.LayoutParams

(LayoutParams.MATCH\_PARENT, LayoutParams.WRAP\_CONTENT);

hintLp = new LinearLayout.LayoutParams

(LayoutParams.WRAP\_CONTENT, LayoutParams.WRAP\_CONTENT);

lEdit = new LinearLayout.LayoutParams

(LayoutParams.MATCH\_PARENT, LayoutParams.WRAP\_CONTENT);

lcheckBox.setMargins(0, 2, 0, 0);

lEdit.setMargins(0, 10, 8, 10);

setContentView(initialize());

}

private ScrollView initialize() {

ScrollView scroll = new ScrollView(getApplicationContext());

scroll.setLayoutParams(lp);

root = new LinearLayout(getApplicationContext());

root.setLayoutParams(lp);

root.setOrientation(LinearLayout.VERTICAL);

Question[] array = parseQuestions();

showView(array);

scroll.addView(root);

return scroll;

}

private void showView(Question[] array) {

lp.width = LayoutParams.MATCH\_PARENT;

for (int i = 0; i < array.length; i++) {

TextView t = new TextView(getApplicationContext());

t.setText(array[i].Title);

lp.setMargins(0, 10, 0, 0);

t.setLayoutParams(lp);

t.setTextSize(18);

root.addView(t);

if (array[i].Type.equals("Singlechoose")) {

TextView hint = new TextView(getApplicationContext());

hint.setText("[Виберіть один варіант]");

hint.setLayoutParams(hintLp);

hint.setTextSize(12);

root.addView(hint);

RadioGroup rg = new RadioGroup(getApplicationContext());

rg.setTag(array[i].Id);

rg.setLayoutParams(lp);

for (int j = 0; j < array[i].Variants.length; j++) {

RadioButton rb = new RadioButton(getApplicationContext());

rb.setLayoutParams(lcheckBox);

rb.setText(array[i].Variants[j]);

rg.addView(rb);

}

root.addView(rg);

}

if (array[i].Type.equals("Multichoose")) {

TextView hint = new TextView(getApplicationContext());

hint.setText("[Виберіть один чи більше варіантів]");

hint.setLayoutParams(hintLp);

hint.setTextSize(12);

root.addView(hint);

for (int j = 0; j < array[i].Variants.length; j++) {

CheckBox check = new CheckBox(getApplicationContext());

check.setTag(array[i].Id);

check.setLayoutParams(lcheckBox);

check.setText(array[i].Variants[j]);

if (j == 0) check.setChecked(true);

root.addView(check);

}

}

if (array[i].Type.equals("Simple")) {

TextView hint = new TextView(getApplicationContext());

hint.setText("[Напишість свою відповідь]");

hint.setLayoutParams(hintLp);

hint.setTextSize(12);

root.addView(hint);

EditText edit = new EditText(getApplicationContext());

edit.setTag(array[i].Id);

edit.setLayoutParams(lEdit);

root.addView(edit);

}

}

Button finish = new Button(getApplicationContext());

finish.setLayoutParams(lp);

finish.setText("Завершити");

finish.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View v) {

for (int i = 0; i < root.getChildCount(); i++) {

View child = root.getChildAt(i);

if (child.getTag() instanceof String) {

String id = (String) child.getTag();

if (child instanceof CheckBox) {

if (((CheckBox)child).isChecked()) {

String text = (String)((CheckBox)child).getText();

if (!hash.containsKey(id)) {

hash.put(id, text);

}

else {

String tmp = hash.get(id);

hash.remove(id);

hash.put(id, tmp + "~" + text);

}

}

}

if (child instanceof RadioGroup) {

int checkId = ((RadioGroup) child).getCheckedRadioButtonId();

RadioButton rb = null;

for (int j = 0; j < ((RadioGroup) child).getChildCount(); j++) {

RadioButton tmp = (RadioButton) ((RadioGroup) child).getChildAt(j);

if (tmp.getId() == checkId) {

rb = tmp;

break;

}

}

String text = (String)rb.getText();

hash.put(id, text);

}

if (child instanceof EditText) {

String text = ((EditText)child).getText().toString();

hash.put(id, text);

}

}

}

String result = "";

Iterator<String> it = hash.keySet().iterator();

while (it.hasNext()) {

String key = it.next();

String values = hash.get(key);

result += key + "11key11" + values + "11new11";

}

Intent data = new Intent();

data.putExtra("result", result);

setResult(RESULT\_OK, data);

finish();

}

});

root.addView(finish);

}

private Question[] parseQuestions() {

String[] records = questions.split("a1b2c3");

Question[] array = new Question[records.length];

for (int i = 0; i < records.length; i++) {

String[] data = records[i].split("`");

String id = data[0];

String title = data[1];

String type = data[2];

String[] variants = data[3].split("~");

array[i] = new Question(id, title, type, variants);

}

return array;

}

}

BluetoothServer.java:

package com.andrey.pzas091.statisticscollection;

import java.io.IOException;

import java.io.InputStream;

import android.bluetooth.BluetoothServerSocket;

import android.bluetooth.BluetoothSocket;

import android.content.Context;

import android.widget.Toast;

public class BluetoothServer implements Runnable {

private BluetoothServerSocket server;

private String question;

private Context ctx;

private final String KVDivide = "11key11";

private final String RecordDivide = "11new11";

private boolean run = true;

public BluetoothServer(BluetoothServerSocket socket, String question, Context ctx) {

server = socket;

this.ctx = ctx;

this.question = question;

}

public void start() {

Thread listen = new Thread(this);

listen.setDaemon(true);

listen.start();

}

@Override

public void run() {

while (run) {

try {

BluetoothSocket client = server.accept();

InputStream is = client.getInputStream();

if (is.available() > 0) {

byte[] buffer = new byte[is.available()];

is.read(buffer);

String data = new String(buffer);

if (data.equals("request")) {

byte[] sendData = question.getBytes();

client.getOutputStream().write(sendData);

}

else {

addToDataBase(data);

}

}

}

catch (IOException e) {

e.printStackTrace();

}

}

}

public void stop() {

run = false;

try {

server.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

private void addToDataBase(String text) {

String[] records = text.split(RecordDivide);

DataBase db = new DataBase(ctx);

for (int i = 0; i < records.length; i++) {

String[] KV = records[i].split(KVDivide);

String key = KV[0];

String value = KV[1];

db.addAnswer(key, value);

}

db.close();

Toast.makeText(ctx, "Отримано дані!", Toast.LENGTH\_SHORT).show();

}

}

BluetoothPoints.java:

package com.andrey.pzas091.statisticscollection;

import java.util.Set;

import android.app.ListActivity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothDevice;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.content.IntentFilter;

import android.os.Bundle;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.ListView;

public class BluetoothPoints extends ListActivity {

public static BluetoothAdapter mBluetoothAdapter;

private ArrayAdapter<BluetoothDevice> adapter;

private final BroadcastReceiver mReceiver = new BroadcastReceiver() {

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

// When discovery finds a device

if (BluetoothDevice.ACTION\_FOUND.equals(action)) {

// Get the BluetoothDevice object from the Intent

BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

// Add the name and address to an array adapter to show in a ListView

adapter.add(device);

}

}

};

@Override

protected void onDestroy() {

unregisterReceiver(mReceiver);

super.onDestroy();

}

@Override

protected void onListItemClick(ListView l, View v, int position, long id) {

BluetoothDevice device = adapter.getItem(position);

Intent data = new Intent();

MainActivity.device = device;

data.putExtra("device", true);

setResult(RESULT\_OK, data);

finish();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

adapter = new ArrayAdapter<BluetoothDevice>(this, android.R.layout.simple\_list\_item\_1);

setListAdapter(adapter);

Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();

if (pairedDevices.size() > 0) {

for (BluetoothDevice device : pairedDevices) {

adapter.add(device);

}

}

IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION\_FOUND);

registerReceiver(mReceiver, filter);

}

}

Exporter.java:

package com.andrey.pzas091.statisticscollection;

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.text.SimpleDateFormat;

import java.util.Calendar;

import android.annotation.SuppressLint;

import android.os.Environment;

public class Exporter {

private File file;

private FileWriter writer;

@SuppressLint("SimpleDateFormat")

public Exporter(String title) throws IOException {

String sd = Environment.getExternalStorageDirectory().getPath();

Calendar c = Calendar.getInstance();

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String date = df.format(c.getTime()).replace("-", "").replace(":", "").replace(" ", "\_");

String path = sd + "/testdb/export/" + title.replace(" ", "\_") + date + ".xml";

file = new File(path);

if (!file.exists()) {

file.createNewFile();

}

writer = new FileWriter(file);

writer.write("<?xml version=\"1.0\" standalone=\"yes\"?>\r\n");

writer.write("<DataSet>\r\n");

}

public void addRow(String id, String id\_question, String text) {

try {

writer.write(" <Row>\r\n");

writer.write(" <\_id>" + id + "</\_id>\r\n");

writer.write(" <id\_question>" + id\_question + "</id\_question>\r\n");

writer.write(" <answer\_text>" +

text

.replace("&", "&amp;")

.replace("<", "&lt;")

.replace(">", "&gt;")

.replace("\"", "&quot;")

.replace("?", "&euro;")

+ "</answer\_text>\r\n");

writer.write(" </Row>\r\n");

}

catch (IOException e) {

e.printStackTrace();

}

}

public void close() {

try {

writer.write("</DataSet>");

writer.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

OpenFileActivity.java:

package com.andrey.pzas091.statisticscollection;

import java.io.File;

import android.app.ListActivity;

import android.content.Intent;

import android.os.Bundle;

import android.os.Environment;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.ListView;

public class OpenFileActivity extends ListActivity {

private ArrayAdapter<String> adapter;

private String path;

@Override

protected void onListItemClick(ListView l, View v, int position, long id) {

String fileName = adapter.getItem(position);

Intent data = new Intent();

data.putExtra("path", path + fileName);

setResult(RESULT\_OK, data);

finish();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1);

setListAdapter(adapter);

String sd = Environment.getExternalStorageDirectory().getPath();

path = sd + "/testdb/import/";

File f = new File(path);

if (!f.exists()) f.mkdirs();

File[] files = f.listFiles();

if (files != null) {

for(int i = 0; i < files.length; i++) {

File file = files[i];

adapter.add(file.getName());

}

}

}

}

Question.java:

package com.andrey.pzas091.statisticscollection;

public class Question {

public String Id;

public String Title;

public String Type;

public String[] Variants;

public Question() {

}

public Question(String id, String title, String type, String[] variants) {

Id = id;

Title = title;

Type = type;

Variants = variants;

}

}

Test.java:

package com.andrey.pzas091.statisticscollection;

import android.widget.ImageView;

public class Test {

public int Id;

private String title;

private String date;

private ImageView icon;

public Test(int id, String title, String date) {

Id = id;

this.title = title;

this.date = date;

}

public String getTitle() {

return title;

}

public void setTitle(String title) {

this.title = title;

}

public String getDate() {

return date;

}

public void setDate(String date) {

this.date = date;

}

public ImageView getIcon() {

return icon;

}

public void setIcon(ImageView icon) {

this.icon = icon;

}

}

TestActivity.java:

package com.andrey.pzas091.statisticscollection;

import java.io.IOException;

import java.util.UUID;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.bluetooth.BluetoothServerSocket;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;

public class TestActivity extends Activity {

public static Test item = null;

private final int ANSWER\_REQUEST = 10;

private final int REQUEST\_ENABLE\_BT = 15;

private TextView tvTestTitle;

private TextView tvTestDate;

private BluetoothServer server = null;

private BluetoothAdapter mBluetoothAdapter;

private final String KVDivide = "11key11";

private final String RecordDivide = "11new11";

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

tvTestTitle = (TextView)findViewById(R.id.tvTestTitle);

tvTestDate = (TextView)findViewById(R.id.tvTestDate);

if (item != null) {

tvTestTitle.setText(item.getTitle());

tvTestDate.setText(item.getDate());

}

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (requestCode == ANSWER\_REQUEST && resultCode == RESULT\_OK) {

if (data.hasExtra("result")) {

String result = data.getStringExtra("result");

addToDataBase(result);

}

}

if (requestCode == REQUEST\_ENABLE\_BT && resultCode == RESULT\_OK) {

startServer();

}

super.onActivityResult(requestCode, resultCode, data);

}

private void addToDataBase(String text) {

String[] records = text.split(RecordDivide);

DataBase db = new DataBase(this);

for (int i = 0; i < records.length; i++) {

String[] KV = records[i].split(KVDivide);

String key = KV[0];

String value = KV[1];

db.addAnswer(key, value);

}

db.close();

}

public void bViewTest\_Click(View view) {

DataBase db = new DataBase(this);

AnswerActivity.questions = db.getQuestions("" + item.Id);

db.close();

Intent intent = new Intent(this, AnswerActivity.class);

startActivityForResult(intent, ANSWER\_REQUEST);

}

public void bExportToXML\_Click(View view) {

Exporter exp = null;

try {

exp = new Exporter(item.getTitle());

}

catch (IOException e) {

e.printStackTrace();

}

DataBase db = new DataBase(this);

db.exportData(exp, item.Id);

db.close();

Toast.makeText(this, "Export succesfull!", Toast.LENGTH\_SHORT).show();

}

public void bStartTest\_Click(View view) {

mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

if (mBluetoothAdapter == null) {

Toast.makeText(getApplicationContext(), "Не знайдено Bluetooth!",

Toast.LENGTH\_SHORT).show();

return;

}

if (server != null) {

Toast.makeText(getApplicationContext(), "Сервер вже працює...", Toast.LENGTH\_SHORT).show();

return;

}

if (!mBluetoothAdapter.isEnabled()) {

Intent discoverableIntent = new

Intent(BluetoothAdapter.ACTION\_REQUEST\_DISCOVERABLE);

discoverableIntent.putExtra(BluetoothAdapter.EXTRA\_DISCOVERABLE\_DURATION, 300);

startActivityForResult(discoverableIntent, REQUEST\_ENABLE\_BT);

}

else {

startServer();

}

}

private void startServer() {

try {

DataBase db = new DataBase(this);

String question = db.getQuestions("" + item.Id);

db.close();

BluetoothServerSocket socket = mBluetoothAdapter.

listenUsingRfcommWithServiceRecord("DataCollectionApp", new UUID(1, 200));

server = new BluetoothServer(socket, question, this);

server.start();

Toast.makeText(getApplicationContext(), "Сервер запущено!", Toast.LENGTH\_SHORT).show();

}

catch (IOException e) {

e.printStackTrace();

}

}

@Override

protected void onDestroy() {

if (server != null) server.stop();

super.onDestroy();

}

}

Продовження програмного коду міститься на диску.

# ДОДАТОК Б

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№ рядка** | **формат** | **Позначення** | **Найменування** | **Кількість** | **Примітка** |
| 1 |  |  |  |  |  |
| 2 |  |  | Документація |  |  |
| 3 |  |  |  |  |  |
| 4 | А4 | КНУ.ПД.06.050103.13.13 | Пояснювальна записка | 92 |  |
| 5 |  |  |  |  |  |
| 6 |  |  | Графічні матеріали |  |  |
| 7 |  |  |  |  |  |
| 8 | А4 | КНУ.ПД.06.050103.13.13.ФС | Функціональна схема роботи програми реєстрації груп користувачів | 1 |  |
| 9 | А4 | КНУ.ПД.06.050103.13.13.СО | Алгоритм створення опитування | 1 |  |
| 10 | А4 | КНУ.ПД.06.050103.13.13.ПО | Алгоритм проходження опитування на мобільному пристрої  програми | 1 |  |
| 11 | А4 | КНУ.ПД.06.050103.13.13.ПВ | Алгоритм проходження опитування на віддаленому мобільному пристрої | 1 |  |
| 12 | А4 | КНУ.ПД.06.050103.13.13.АВ | Алгоритм аналізу відповідей | 1 |  |
| 13 | А4 | КНУ.ПД.06.050103.13.13.СБ | Структура бази даних | 1 |  |
| 14 | А4 | КНУ.ПД.06.050103.13.13.ПМ | Програма для мобільних телефонів. Зовнішній вигляд | 1 |  |
| 15 | А4 | КНУ.ПД.06.050103.13.13.ПП | Програма для персонального комп’ютера. Зовнішній вигляд | 1 |  |
| 16 |  |  |  |  |  |
| 17 | CD | КНУ.ПД.06.050103.13.13 | Програмні матеріали | 6,9 | Мбайт |
| 18 |  |  |  |  |  |
| 19 | CD/В | КНУ.ПД.06.050103.13.13.ПВ | Вихідні тексти програм | 4,00 | Мбайт |
| 20 | CD/І | КНУ.ПД.06.050103.13.13. ПІ | Інсталяційний пакет | 4,90 | Мбайт |
| 21 | CD/П | КНУ.ПД.06.050103.13.13.ПП | Презентація системи | 1 | Мбайт |