РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота магістра містить: 68с, 9рис. 18джер.

РОЗРОБКА ДОДАТКУ, КРОСПЛАТФОРМЕНІ ТЕХНОЛОГІЇ, ТЕХНОЛОГІЯ XAMARIN, МОВА ПРОГРАМУВАННЯ C#, XML ТЕХНОЛОГІЯ, МОБІЛЬНІ ПЛАТФОРМИ.

Метою створення кваліфікаційної роботи є розробка та введення в обіг мобільної гри «знайти пару» за допомогою кросплатформеної технології XAMARIN яка дозволяє в зручний спосіб реалізувати програмні продукти.

Методи розробки базуються на інструментах розробки технології XAMARIN. Платформа Xamarin є рівень абстракції, який забезпечує управління взаємодією між загальним кодом і кодом базової платформи.

В результаті роботи розглянуто методи різні методи та технології розробки , обґрунтовано доцільність використання вибраної технології, описано предметну область. Розроблено серверну частину програми та клієнтську частину для середовища Android та Windows.

APPLICATION DEVELOPMENT, CROSSPLATFORM TECHNOLOGIES, XAMARIN TECHNOLOGY, C # PROGRAMMING LANGUAGE, XML TECHNOLOGY, MOBILE PLATFORMS.

The purpose of the qualification work is to develop and put into circulation a mobile game "find a pair" using cross-platform technology XAMARIN, which allows you to easily implement software products.

Development methods are based on XAMARIN technology development tools. The Xamarin platform is an abstraction layer that provides control over the interaction between the common code and the base platform code.

As a result of work methods of various methods and technologies of development are considered, expediency of use of the chosen technology is substantiated, the subject area is described. The server part of the program and the client part for the Android and Windows environment have been developed.

Зміст

Вступ.

Популярність мобільних засобів появилася через велику взаємодію з людино так як мобільні пристрої за допомогою сучасних технологій здатні в великому обсязі задовольнити інформаційні потреби людей. Через свою зручність та гнучкість вони замінили багато пристроїв, що були необхідними для користувача та колись здавалися незамінними, такі як стаціонарні телефони, відеокамера, навігатор, радіо приймач, фотокамера, блокнот, календар, відеокамера. Також дані пристрої зайняли і велику нішу і сфері відео ігор. Ні для кого не секрет, що відео ігри міцно зайняли свою позицію в сучасній індустрії розваг. Існують спроби виділити комп'ютерні ігри як окрему область мистецтва, поряд з театром, кіно і т.п. Розробка ігор може виявитися не тільки захоплюючим, але й прибутковою справою. А в зв'язку з ростом кількості мобільних пристроїв і уніфікації способів розробки додатків для кожного з них, з'являється широка перспектива комерційної розробки. Ринок мобільних пристроїв, а відповідно і їх користувачів стрімко зростає і потрібно все більше і більше контенту для них: програми, ігри, утиліти, спрямовані на повне розкриття потенціалу мобільних пристроїв. Цим і визначається актуальність обраної теми. Метою роботи є розробка кроссплатформенного клієнтсерверного ігрового програми «Знайди пару». Об'єктом дослідження є технологія кроссплатформенной розробки на базі XAMARIN, що дозволяє розробляти Кросплатформені додатки в середовищі MS Visual Studio на мові C# і компілювати їх код для кожної платформи. Предметом дослідження є проектування переносний кроссплатформенной бібліотеки класів, розробка серверної частини, інтерфейсу на платформах Windows і Android. Практична новизна: досліджувана технологія абсолютно нова і стрімко розвивається, з кожним місяцем яка приваблює все більшу кількість розробників. Тим самим наближаючи і об'єднуючи безліч існуючих платформ воєдино для зручної, швидкої та якісної розробки додатків. Практична значимість результату виконаної роботи: розроблене додаток можна використовувати в комерційних

цілях, як для продажі, так і заробітку на рекламі (в перспективі зростання кількості користувачів).

1. Середовища розробки і Технологія XAMARIN

1.1 огляд існуючих середовищ розробки

1.1.1 Інтегроване середовище розробки Eclipse

Інтегроване середовище розробки Eclipse Eclipse - це розширювана, open-source інтегроване середовище розробки. Спільно з Java Development Tools, платформа Eclipse надають безліч різних можливостей, які можна спостерігати в комерційних середовищах розробки: підсвічування синтаксису в редакторі, компіляція коду, відладчик рівня вихідного коду з підтримкою "ниток" (threads), навігатор по класах, файловий менеджер і менеджер проектів, інтерфейси для стандартних контролюючих систем вихідного коду, таких як, наприклад, CVS і ClearCase. Крім цього Eclipse містить ряд унікальних можливостей, наприклад, рефакторинг коду, автоматичне оновлення і збірка коду, список завдань, підтримка можливості тестування модулів за допомогою JUnit, а також інтеграція з інструментом збірки додатків Jakarta Ant. [1,2] Незважаючи на велику кількість стандартного набору можливостей, Eclipse відрізняється від традиційних середовищ розробки по декількох фундаментальних особливостей. Найцікавіша можливість Eclipse - це абсолютна нейтральність щодо платформи і мови програмування. До того ж до еклектичному набору мов програмування, які підтримуються Eclipse Consortium (Java, C / C ++, Cobol), існує безліч сторонніх проектів, за допомогою яких можна забезпечити підтримку цікавить мови програмування в Eclipse. На сьогоднішній день існують реалізації наступних популярних мов програмування: Python, Eiffel, PHP, Ruby, і C #. [1,2] Платформа Eclipse надається, завдяки Eclipse Consortium, у вигляді заздалегідь скомпільованих виконуваних файлів для Windows, Linux, Solaris, HP-UX , AIX, QNX, і Mac OS X. Дуже багато уваги концентрується навколо архітектурної системи plug-in'ов

ції платформи, а також 12 "багатих" API (Application Programming Interface), що поставляються з Plug-in Development Environment для розширення Eclipse.

Додати підтримку нового типу редактора або мови програмування дуже просто, завдяки добре спроектованим API і будівельним блокам, які надає Eclipse. [1,2]

1.1.2 Інтегроване середовище розробки NetBeans

NetBeans є офіційною середовищем розробки для Java 8, надає всі засоби, необхідні для створення професійних десктоп додатків, корпоративних, мобільних і веб-додатків. За допомогою його редакторів, аналізаторів і перетворювачів коду, розробник може легко і швидко оновлювати існуючі програми, щоб використовувати нові мовні конструкції Java 8, таких як лямбда-виразів, функціональних операцій, і т.д .. [3,4] Пакетні аналізатори та перетворювачі коду дозволяють виконувати пошук по декільком додаткам одночасно, порівнюючи існуючі структури коду для перетворення їх в нові мовні конструкції Java 8. [3,4] Робоча область середовища є повністю настроюється - існує можливість для користувача настройки дій, які виконуються за допомогою панелі, призначення "гарячих "клавіш і т.д. Так само середовище має в своєму складі розширений багатомовний редактор для різних мов програмування - Java, C / C ++, Ruby, Groovy, PHP, JavaScript, CSS, XML, HTML, RHTML, JSP, документацію Javadoc. Існує можливість розширення функцій редактора з метою підтримки будь-якого іншого мови. [3,4] Редактор NetBeans робить відступи рядків, перевіряє відповідність дужок і слів, підсвічує синтаксис вихідного коду. Проводиться перевірка помилок під час введення, відображення варіантів для автозавершення коду і фрагментів документації по вашій мові програмування. Для прискорення розробки середовище може генерувати і вставляти в вихідний код стандартні фрагменти коду на Java або іншими мовами. [3,4] 13 1.1.3

1.1.3 Інтегроване середовище розробки Visual Studio 2015 Enterprise

Visual Studio 2015 Enterprise - це інтегроване середовище розробки, яка володіє всім необхідним для створення відмінних додатків для мобільних пристроїв, настільних додатків, веб-додатків і хмарних рішень. Дозволяє писати код для iOS, Android і Windows в одній інтегрованому середовищі розробки. Надає доступ до зручної і функціональної середовищі IntelliSense, пріоритетами простий навігації по коду, швидкої збірки і розгортання в найкоротші терміни. [5,9] За допомогою інструментів діагностики Visual Studio 2015 Enterprise і IntelliTrace можна переглядати журнал виконання коду і переходити до перевіряється станом без завдання точок зупину вручну. Це заощадить чимало часу, яке доцільно витратити більш продуктивно. [5] Visual Studio 2015 Enterprise спрощує інтеграцію навантажувального тестування в процес розробки і допомагає уникнути небезпечних сюрпризів в робочому середовищі. Причиною цих сюрпризів може бути поширення по всьому світу, масштаб бази клієнтів або проблеми, які проявляються тільки при багатоденних циклах виконання, - тестування навантаження Visual Studio 2015 надасть аналітичні дані для усунення цих проблем ще до розгортання. [5] Visual Studio 2015 Enterprise включає функції Xamarin для розробки додатків для Android, iOS і Windows. Дозволяє створювати додатки з продуктивністю машинного коду і машинним призначеним для користувача інтерфейсом за допомогою інструментів корпоративного рівня. Полегшує оптимізацію продуктивності додатка за допомогою даних профілювання, а так само дослідження їх в середовищі виконання для швидкого пошуку помилок. Дозволяє перевірити якість розробки на справжніх пристроях в тестовому хмарі Xamarin. [5,7]

1.2 Обґрунтування вибору середовища розробки

Оскільки основним досліджуваним мовою програмування в процесі навчання є C #, а його «рідна» середовище розробки - Visual Studio і платформа .NET, середовищем розробки слід вибрати VisualStudio 2015 Enterprise з розширенням XAMARIN. Це дозволить використовувати всю міць платформи .NET в поєднанні з новими вбудованими інструментами та бібліотеками для кроссплатформенной розробки для всіх існуючих платформ (в тому числі універсальних програм для Windows 10 і Windows Phone), забезпечить зручність розробки при використанні звичних інструментів і допоміжних утиліт. Крім цього вибір обумовлений наявністю учнівської ліцензією за передплатою DreamSpark для студентів вищих навчальних закладів. Інтегроване середовище розробки Eclipse є спочатку спрямованої на розробку на Java, C / C ++, Cobol, хоч і дозволяє розробляти програми на мові C #, підключивши спеціальне розширення, вона все одно поступається за можливостями VisualStudio 2015 Enterprise з розширенням XAMARIN. Інтегроване середовище розробки NetBeans є рідною середовищем для розробки на Java і основний акцент в ній зроблено на розробку саме цією мовою, так само за замовчуванням середовище не підтримує мову C #.

1.3 Технологія кроссплатформеної розробки XAMARIN

1.3.1 Об'єкт дослідження. Об'єктом в даній роботі є технологія кроссплатформеної розробки на базі XAMARIN.

Існує два основні підходи до розробки мобільного платформ на C #, які ми можемо використовувати окремо, по черзі або в тандемі:[18]

• Xamarin.Forms - це багатоплатформовий інструментарій для користувача інтерфейсу для Android, iOS і Windows Phone.

• Платформний призначений для користувача інтерфейс використовує Xamarin.Android, Xamarin.iOS і Windows Phone SDK.

Xamarin.Forms містить повністю багатоплатформовий інструментарій, що надає єдиний набір елементів управління призначеним для користувача інтерфейсом, макетів і сторінок, які вміло зіставляються з відповідними рідними прив'язками призначеного для користувача інтерфейсу на iOS, Android і Windows Phone. Так як Xamarin.Forms новіше платформних бібліотек, він також менш повнофункціональний. Кожен реліз наближає нас до повнофункціонального крос-платформенному добру, але іноді нам потрібно більше, ніж Класи Xamarin.Forms можуть бути запропоновані з коробки. У цих випадках ми використовуємо бібліотеки, специфічні для даної платформи, або для всієї сторінки, або тільки для її частин, використовуючи призначені для користувача рендерер Xamarin.Forms, звані PageRenderers. Підхід, орієнтований на конкретну платформу, є більш старим і усталеним і, отже, досить деталізованим і повнофункціональним. Він включає в себе бібліотеки, які прив'язуються безпосередньо до переносних специфічним API: Xamarin.Android для Android, і Xamarin.iOS для iOS. Для Windows Phone ми використовуємо Windows Phone SDK, рідний API, який не потребує прив'язки Xamarin. Ці специфічні для платформи бібліотеки дають нам глибокий доступ до нативним

призначеним для користувача інтерфейсів для надання візуально приголомшливий, інтерактивно насичений користувальницький досвід. Це коштує дорого: код, специфічний для платформи, вимагає окремого проекту призначеного для користувача інтерфейсу для кожної платформи з невеликою кількістю повторного використання коду. Також ця платформа дозволяє:

• Мобільний призначений для користувача інтерфейс - це найбільша область нового навчання для мобільних розробок на C #. Xamarin.Forms надає крос-платформний інструментарій для користувача інтерфейсу, що містить готові форми, сторінки, макети, уявлення і елементи управління. Xamarin.iOS і Xamarin.Android забезпечують прив'язки до своїх рідних призначеним для користувача інтерфейсів.[18]

• Рівень доступу до даних в мобільному додатку зазвичай пов'язує елементи управління і сторінки з моделями даних, заповненими з локальної бази даних, яка синхронізована з віддаленим сервером даних за допомогою веб-служб[18]

• Доступ до локальних баз даних через SQLite - це зміна в порівнянні зі звичайними виробниками баз даних, хоча доступ ADO.NET забезпечує знайомий вхід, а компонент SQLite.NET - це функціональна можливість. (Див.

• Прив'язка даних займає центральне місце в розробці Xamarin.Forms і часто виконується з використанням шаблону Model-View-ViewModel (MVVM).

• Крос-платформна архітектура являє собою набір стратегій спільного використання коду для досягнення нашої мети написати один раз, запустити в будь-якому місці. До них відносяться переносні бібліотеки класів (PCL) з впровадженням залежностей, загальні файли і проекти, а також умовна компіляція.

У нас є величезна кількість нового матеріалу, який ми можете поглинути при переході від розробки полотна до Кросплатформені мобільні розробки, в основному в області користувальницького інтерфейсу. Вперше виходить новий

UI API операційної системи. Xamarin допомагає в цьому, надаючи платформні прив'язки на C # до основних ОС з Xamarin.Android і Xamarin.iOS, в той час як Xamarin.Forms надає Кросплатформені прив'язки до обох з них, а також до Windows Phone. Інший комплекс проблем пов'язаний з відмінностями в дизайні між веб-додатками і мобільними додатками. Компактний екран, сенсорна чутливість і портативний форм-фактор . Принципово новий користувальницький досвід (UX). Це вимагає принципово нового підходу до проектування та розробки, що приводить нас до вивчення дизайну мобільного користувальницького інтерфейсу.[18]

Методи призначеного для користувача інтерфейсу складають ядро більшості розробок програмного забезпечення для мобільних пристроїв. Нинішні апаратні обмеження малих пристроїв спонукають нас залишити важку навантаження на ПК і сервери в дальньому кінці нашої мережі. послуги. Більшість компонентів, які працюють в мобільному бізнес-додатку, підтримують видимий призначений для користувача інтерфейс. Мобільний бізнес і рівні доступу до даних часто є скороченими версіями своїх побратимів на серверній стороні. Це означає, що те, що нам потрібно найчастіше є компоненти користувальницького інтерфейсу, щоб допомогти нам проектувати екрани з використанням макетів, реалізувати елементи управління для введення і відбору даних, створювати списки і таблиці для відображення і редагування даних, створювати для користувача навігацію, а також використовувати зображення для фону і іконки. Теми UI в цій книзі охоплюють функції, найбільш часто використовувані при розробці мобільних додатків. У кожному розділі ми починаємо з найпростіших, найбільш кроссплатформенних підходів, а потім заглиблюємося в специфіку платформи для деталізації і деталізації. Такі теми мобільного користувальницького інтерфейсу, освітлені в цій книзі:

• Екрани, уявлення або сторінки схожі з еквівалентами веб-сторінок і робочого столу на C #, з використанням елементів управління з методами і властивостями, а також з подіями обстрілу, які ми обробляємо в наших контролерів.

• Макети допомагають нам організувати позиціонування і форматування елементів управління, що дозволяє структурувати і оформляти екрани нашого мобільного застосування. (Див. Главу 3.)

• Засоби управління полегшують взаємодію з користувачем і введення даних, що є унікальним для мобільного користувальницького інтерфейсу і істотно відрізняється від інтерфейсу ПК / миші, в основному за рахунок використання жестів.

• Списки є одним з найпотужніших методів відображення та відбору даних в мобільних додатках.

• Навігація дозволяє користувачеві переміщатися по додатком, переміщатися з екрану на екран і отримувати доступ до функцій. Ієрархічна навігація, модальні екрани, навігаційні ящики, оповіщення, блискучі списки і інші ключові шаблони складають ядро навігації мобільного користувальницького інтерфейсу.

• Управління станом - це обробка даних, що передаються між екранами при навігації користувача по додатку.

• Зображення займають центральне місце в мобільній роботі, в меню, списках, сітках, каруселях та інших макетах.

1.3.2 Методи дослідження.

Для дослідження об'єкта використовуються наступні методи:

1) огляд існуючих цільових платформ

2) огляд методів портиртування додатки на різні платформи

Ці методи є основними, оскільки досліджувана технологія спрямована саме на кроссплатформені розробку на мові C #. Необхідно вибрати цільові платформи, під які буде проводитися розробка, а так само можливі методи перенесення розробленого додатка на інші платформи в майбутню перспективу. Оскільки розробляється програма має мати архітектуру клієнт-сервер, це

дозволить більш глибоко зануриться в технологію при розробці, зокрема вивчити її можливості для роботи з мережею. В якості цільових платформ слід звернути увагу на Windows і Android. Оскільки вони найбільш поширені і для написання програми для них, не потрібно купувати додаткове обладнання і ліцензію. Як наприклад, для того щоб скомпілювати програми для iOS, необхідно володіти MAC сумісним комп'ютером і установленим на ньому середовищі розробки X-CODE, що в свою чергу вимагає відчутних матеріальних витрат. А для розробки для Windows і Android, досить мати комп'ютер під керуванням ОС Windows і установленим середовищем розробки VisualStudio, учнівської ліцензії для яких досить, щоб вирішити всі поставлені завдання випускної кваліфікаційної роботи. Методи портірвоанія в основному визначає архітектура програми. Самі розповсюдженні методи проектування програми із загальним кодом: рішення із загальним проектом і рішення з переносною бібліотекою класів. Рішення із загальним проектом має на увазі реалізацію в одному рішенні проектів інтерфейсу, поведінки, моделей даних і т.д. Для портування такого додатка потрібно додати в рішення проект інтерфейсу необхідної цільової платформи і використовуючи реалізовані моделі даних і поведінки реалізувати нову версію програми. Рішення з переносною бібліотекою класів на увазі відділення моделей даних і поведінки в окрему динамічну бібліотеку і в Надалі підключати її до нових реалізацій під конкретні платформи. Найоптимальнішим варіантом для даної розробки буде використовувати рішення з переносною бібліотекою класів, оскільки вона буде містити моделі даних і поведінки, які використовуються на сервері і клієнтів, а так само забезпечить простоту портування під інші платформи.[18]

Призначені для користувача рендери дозволяють нам заглиблюватися в порівнянні з готовими елементами управління призначеним для користувача інтерфейсом Xamarin.Forms і використовувати переваги платформного набору функцій для користувача інтерфейсу, зберігаючи при цьому крос-платформний підхід. Додатки Xamarin.Forms за своєю природою є кроссплатформенную і працюють на всіх трьох основних платформах, використовуючи одну і ту ж

кодову базу. Це добре працює для базових конструкцій і використання певних елементів управління. Тим не менш, у багатьох проектах буде розвиватися потреба в більш глибокому використанні призначеного для користувача інтерфейсу, наприклад, нюанси дизайну на одному елементі управління, нативні модальні діалогові вікна, додаткова графіка або анімація на сторінці, або будь-які вимоги, які виходять за рамки того, що надає Xamarin.Forms в поточному релізі. Це досягається за рахунок підкласифікації рідних елементів управління і реалізації PageRenderers для створення призначених для користувача елементів управління, які дають повний доступ до переносних залежною функціональності призначеного для користувача інтерфейсу, використовуючи Xamarin.iOS і Xamarin.Android. Ці специфічні для платформи елементи управління можуть бути використані в Xamarin.[18]

Мобільний рівень доступу до даних відходить від конструкцій, до яких ми звикли в веб-додатках, і ближчий до тих, які можна знайти в настільних додатках. Підходи варіюються по складності від популярного шаблону MVVM до Model-View-Controller (MVC) і базових CRUD (Create / Insert / Update / Delete). Пов'язані з даними сторінки, як правило, надходять в локальну базу даних пристрою, яка синхронізується з віддаленим сервером даних за допомогою веб-служб.[18]

Веб-сервіси в мобільній розробці на C # є основоположним аспектом повторного використання коду. Багато з цих сервісів залишаються такими ж, як і для мобільних додатків, до яких ми звикли при створенні веб-додатків. Проте, мобільні веб-сервіси більш схожі на ті, які можна знайти в настільних додатках, що відрізняються від веб-сервісів додатків в першу чергу важливістю ніж синхронізувати в автономному режимі. Інтерфейси створення, оновлення та видалення відображаються в режимі онлайн для RESTful дзвінків з безлічі платформ і пристроїв. Самі справедливі з незайманих мешкають тут в шаблонах мобільних веб-сервісів, ідеальне місце для кроссплатформенного, скриптового коду.[18]

Серверний компонент веб-сервісів залишається незмінним для мобільних додатків, в порівнянні з тим, до чого ми звикли з веб-додатками, за винятком додавання синхронізації даних з локальними мобільними сховищами даних як для онлайн, так і для оффлайн-використання. Використання в автономному режимі вимагає, щоб базовий набір даних зберігався під рукою в локальній базі даних і синхронізуватися при наявності з'єднання. Не всі програми підтримують використання в автономному режимі.[18]

SQLite є стабільним і надійним кросплатформним продуктом для розробки мобільних баз даних, проектом з відкритим вихідним кодом, який працює на iOS, Android і Windows пристроях. Xamarin рекомендує його по відношенню до ряду інших проектів сторонніх розробників, і саме про нього розповідається в цій книзі. Xamarin забезпечує доступ і створення баз даних SQLite в середовищі розробки, надає ADO.NET з підтримкою SQLite. Також існує компонент SQLite.NET Xamarin, обгортка на C # навколо рівня даних SQLite на C, що забезпечує низько рівневий доступ до бази даних SQLite, яка включає в себе асинхронні транзакції. Все це дозволяє легко і безболісно підключатися до бази даних, створювати і індексувати таблиці, а також читати і записувати рядки.[18]

Прив'язка даних є послідовною і крос-платформної при використанні Xamarin.Forms. Модельований після прив'язки даних в Windows Presentation Foundation (WPF), шаблон MVVM є центральним для його реалізації. У коді ми прив'язуємо контрольні поля до нашої моделі даних і механізм прив'язки є автоматичним. Ручна реалізація події PropertyChanged дозволяє коду синхронізуватися з джерелом даних. Прив'язка виконується в коді або в розширювана мова розмітки додатків XAML і може бути одностороннім або двостороннім. Елементи управління, списки і текст прив'язані до джерела даних або до властивостей один одного. Все більше число сторонніх виробників надають Xamarin. Форми управління пакетами, які включають прив'язані до даних графіки та сітки, такі як Телерік, Інфрагістіка, Синхронізація і DevExpress. Прив'язка даних не вбудована в Xamarin.Android і Xamarin.iOS.

Платформні реалізації прив'язки даних зазвичай досягаються за допомогою сторонніх бібліотек з відкритим вихідним кодом, таких як MvvmCross і MVVM Light.

Точно так же, як .NET забезпечив об'єднує інфраструктуру, що охоплює багато системи та мови, Xamarin заповнює прогалини між мобільними операційними системами і відповідними мовами розробки: iOS і Objective-C, Android і Java, а також Windows Phone і планшетними комп'ютерами і C #. Таким чином, Xamarin розширює .NET в область мобільних пристроїв, виходячи далеко за рамки операційних систем Windows. Крім того, що це надзвичайно круто, реальною цінністю тут є можливість обміну і повторного використання коду між проектами і платформами і між ними. Найбільші переваги інструментів Xamarin можна знайти в кроссплатформенную коді, тому багатоплатформовий підхід до мобільних шаблонами дасть найбільшу віддачу. Інструменти Xamarin надали нам можливість один раз побачити єдинорога і розгорнути його в будь-якому місці.

Найбільший ворог, з яким ми стикаємося в нашому прагненні до кроссплатформенной реалізації, - це код, специфічний для платформи. Цей код повинен бути реалізований по-різному в залежності від платформи, будь то iOS, Android або Windows Phone.

Крос-платформні моделі однакові незалежно від операційної системи. Крос-платформний код іноді називають загальним кодом, або основним кодом, так як він розподіляється між проектами для різних мобільних операційних систем.

Xamarin.Forms вирішує найскладнішу крос-платформену завдання: призначений для користувача інтерфейс. Розробники, що використовують Xamarin, мають повністю кроссплатформне рішення для даних, яке є назвою доступ до даних за допомогою ADO.NET або SQLite.NET з використанням SQLite, а потім веб-сервісів. Тим не менш, завжди буде існувати код, специфічний для платформи, як показано нижче:

• В інтерфейсі

• Функціональні можливості конкретного пристрою, такі як камера і місце розташування

• Графіка і анімація

• Дозволи на безпеку, файли і пристрої

Після того, як ми визначили специфічний для платформи і багатоплатформовий код, питання полягає в тому, як організувати його в кроссплатформенную архітектуру.

1.4 Опис предметної області

Розробляється гра спрямована на розвиток зорової пам'яті та дрібної моторики пальців, які розвиваються в процесі гри. Для цього гравцеві пропонується проявити ряд картинок по парам. Ігрове поле являє собою 16 і 36 клітин для нормального і важкого рівня складності відповідно. У кожній клітині випадковим чином розташована одна з 8 (нормальний рівень складності) і 18 (важкий рівень складності) пар картинок. Кожна картинка прихована від гравця. Гравцеві належить відкрити всі клітини з картинками якомога швидше. Ігровий процес представляється в такий спосіб: гравець вибирає клітку, йому показується приховане зображення, потім вибирає другу клітку і якщо зображення на обох клітинах збігаються, то вони залишаються відкритими, і гравець вибирає далі. Якщо зображення не збігаються, то через деякий час вони ховаються і гравцеві належить знову шукати пари. Гра закінчується, як тільки гравець відкриє всі клітини з картинками. За результатом час проходження заноситься в таблицю рекордів.

1.5 Технічне завдання

1.5.1 Введення

1.5.1.1 Найменування програми

Найменування гри - «Знайди пари».

1.5.1.2 Коротка характеристика області застосування

Програма призначена для здійснення ігрової діяльності в вільний час на різних пристроях.

1.5.1.3 Цілі і терміни розробки

Метою розробки є удосконалення практичних навичок розробки кроссплатформенних клієнт-серверних додатків. Терміни розробки: 2 квартал 2020 року.

1.5.2 Підстави для розробки

1.5.2.1 Підстава для проведення розробки

Підставою для проведення розробки є завдання на розробку ігрового програми, в рамках підготовки випускної кваліфікаційної роботи на здобуття ступеня бакалавра. Що є частиною навчального курсу для студентів ТНТУ кафедри ФІС навчаються за напрямом Програмна інженерія.

1.5.2.2 Найменування та умовне позначення теми розробки

Найменування теми розробки - «Розробка гри« Знайди пари »».

1.5.3 Вимоги до програми або програмного виробу

1.5.3.1 Вимоги до ігрового поля

Для нормального рівня складності ігрове поле складається з 16 клітин, у вигляді квадрата зі стороною 4 клітини. Кожна клітина зберігає одну з 8 пар картинок. Для важкого рівня складності ігрове поле складається з 36 клітин, в вигляді квадрата зі стороною 6 клітин. Кожна клітина зберігає одну з 18 пар картинок. За замовчуванням картинки приховані від гравця.

1.5.3.2 Вимоги до складу виконуваних функцій

Програма повинна забезпечувати можливість виконання перерахованих нижче функцій:

1) функція автоматичної розстановки картинок на ігровому полі;
2) функція вибору складності гри: нормальний і важкий рівень;
3) функція збереження результатів гри в файлі;

4) функція відображення ранжируваних результатів попередніх ігор;

5) функція збереження результатів гри в глобальному сховище на сервері;

6) функція отримання результатів інших гравців з сервера.

1.5.3.3 Вимоги до логіки гри

Гра повинна функціонувати за такими правилами:

1) відкрити одночасно можна не більше двох клітин;

2) клітини відкриваються кліком (киснем) по клітці;

3) якщо відкриті клітини мають різні картинки, то через 500 мс вони приховують свої картинки;

4) якщо відкриті клітини мають однакові картинки, то вони залишаються відкритими до кінця гри;

5) поки не сховаються дві відкриті різні картинки, відкривати інші не можна;

6) гра закінчується після відкриття всіх клітин.

1.5.3.4 Вимоги до цільової платформі розгортання

Основна логіка гри і клієнтська частина взаємодії з сервером повинна бути сумісною з наступними платформами Windows, Windows Phone, Android, iOS. Інтерфейс розробляється з урахуванням особливостей кожної платформи індивідуально.

1.5.3.5 Вимоги до версії платформи

Для Windows - версія 10, мінімальна збірка 10240;

Для Android - версія 5.0.1;

Для Windows Phone - версія 8.1;

Для iOS - версія 7.

1.5.3.6 Вимоги до сервера рекордів

Сервер повинен являти собою REST-сервіс, який забезпечує доступ клієнтів до глобального сховища рекордів. Надає можливість збереження свого рекорду, а також отримання відомостей про рекорди інших гравців.

1.5.3.7 Вимоги до інтерфейсу користувача

Програма повинна забезпечувати взаємодію з користувачем (Оператором) за допомогою графічного призначеного для користувача інтерфейсу, розробленого відповідно до рекомендацій компанії-виробника операційної системи.

1.5.4 Вимоги до складу і параметрів технічних засобів

Пристрій, на якому планується виконати розгортання цього додатка, має відповідати технічним вимогам, що пред'являються виробником ОС до апаратної частини технічного засобу.

1.5.5 Вимоги до вихідного коду і мов програмування

Тексти програм програми повинні бути реалізовані на мові C #. В якості інтегрованого середовища розробки програми повинна бути використане середовище MS Visual Studio 2015 Enterprise с XAMARIN.

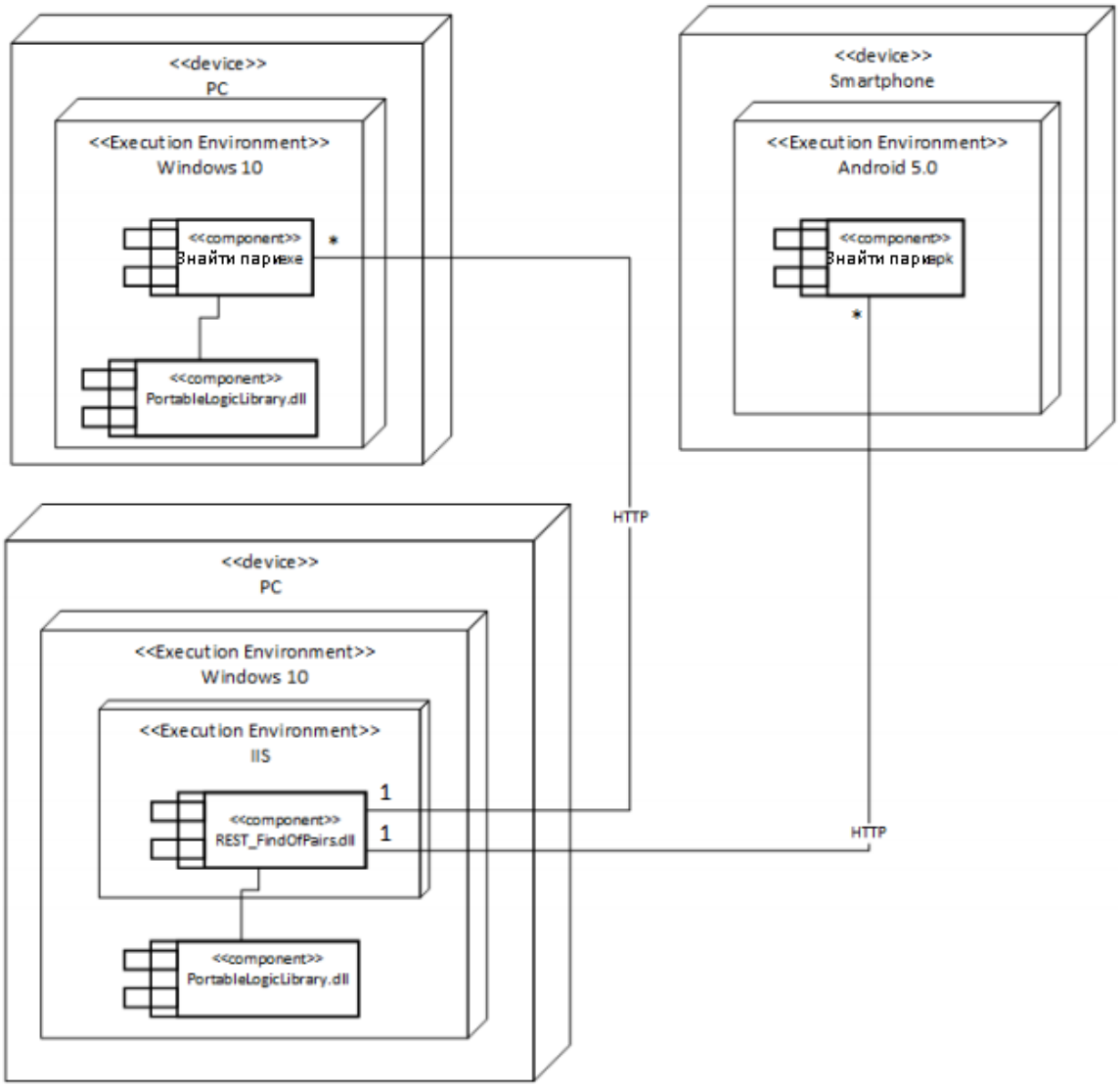
1.5.6 Вимоги до програмних засобів, які використовуються програмою

На комп'ютерах під управлінням Windows 10 необхідна обов'язкова наявність встановленого пакета MS .NET FrameWork не нижче версії 4.5

2 ПРОЕКТУВАННЯ додатків «ЗНАЙДИ ПАРУ»

Кроссплатформенна розробка має на увазі створення додатка для виконання на кількох платформах. Оскільки бізнес-логіка програми не повинна відрізнятися від конкретної платформи, то при перенесенні додатки з однієї платформи на іншу, буде відбуватися дублювання великої кількості коду, за винятком реалізації інтерфейсу і деякі інші особливості роботи з даними, таймерами і ін. (різні простору імен, різні імена класів, різний механізм виконання затримок і ін.). Тому доцільно відокремити бізнес-логіку від інтерфейсу, тим самим полегшуючи розробку і подальшу підтримку програми. При будь-яких змінах і доповненнях у бізнес-логіці не потрібно виправляти всі версії додатка для кожної платформи, а достатньо буде скорегувати код в бібліотеці бізнес-логіки. Для реалізації бізнес-логіки доцільно використовувати переноситься бібліотеку класів. Вона дозволяє використовувати можливості, однаково доступні на всіх цільових платформах і не дозволяє використовувати будь-які нативні функції конкретної платформи. Тим самим просто використовуючи переноситься бібліотеку класів з початковим зазначенням цільових платформ, вже забезпечується обмежене коло доступних класів, на основі яких реалізується вся бізнес-логіка. Так само переноситься бібліотека класів дуже зручна при розробці кроссплатформенних додатків, у міру необхідності випуску програми під чергову цільову платформу досить підключити бібліотеку і реалізувати інтерфейс.

Оскільки що розробляється є кросплатформним з архітектурою клієнт-сервер, доцільно винести в переноситься бібліотеку класів не тільки бізнес-логіку, а й модель даних, а так само реалізацію процесу взаємодії з сервером. В цілому структуру взаємодії всіх компонентів відображає UML діаграма розгортання, в Відповідно до малюнком 1.



Малюнок 1 - UML діаграма розгортання

У ролі сервера виступає REST сервіс, який дозволяє зберігати і управляти глобальним списком рекордів всіх гравців. Клієнт, використовуючи API методи сервісу, може отримати всі встановлені рекорди, отримати список рекордів певної складності, відправити свій рекорд.

У ролі клієнтів виступають клієнтські версії гри для платформ Windows і Android. На кожному пристрої використовується переноситься бібліотека класів PortableLogicLibrary.dll, яка в Windows є окремим файлом і підгружається під час виконання, а в Android вона впроваджується в виконуваний файл при компіляції.

2.1 Стерпна бібліотека класів PortableLogicLibrary

Бібліотека PortableLogicLibrary.dll включає в себе три простору імен:

1) Server - містить класи для взаємодії з сервером;
2) Score - містить класи, що описують модель даних результатів гри і процедуру їх підрахунку;
3) Logic - містить класи логіки гри.

2.1.1 Простір імен Server

Містить один клас RestClient, який реалізує всі необхідні методи для доступу до REST сервісу з глобальним сховищем рекордів.

Клас включає два властивості і п'ять методів:

1) public int KeyAccess - властивість повертає цілочисельне значення ключа доступу до сервера;
2) public string ServerURI - властивість повертає рядок URI сервера;
3) public async Task <Records> GetRecords () - асинхронний метод повертає об'єкт Records, що містить всі глобальні рекорди;
4) public async Task <Records> GetRecords (int id) - асинхронний метод повертає об'єкт Records, що містить всі рекорди гравця з номером id;
5) public async Task <IEnumerable <Player >> GetRecordsAsDifficulty (Difficulty dif) - асинхронний метод повертає об'єкт <IEnumerable <Player>, що містить список рекордів заданої складності dif;
6) public async Task <bool> PostPlayer (Difficulty dif, Player player) - асинхронний метод відправляє об'єкт player, який містить відомості про встановленому рекорд, параметр dif визначає, на який складності був встановлений рекорд. Метод повертає true, якщо рекорд успішно був відправлений на сервер, і false, якщо відправка зазнала невдачі;

7) public async Task <bool> Delete (Difficulty dif, Player player) - асинхронний метод видаляє з сервера рекорд player зі списку рекордів в залежності від зазначеної складності dif, і повертає true у разі успіху і false у випадку невдачі.

2.1.2 Простір імен Score

Включає в себе три класи Records, Player, CalculateRecord і перерахування Difficulty. Класи Records і Player визначають модель даних. перший зберігає всі рекорди, розбиті за складністю, а другий визначає гравця, який встановив рекорд. Клас Records має чотири властивості:

1. public List <Player> Normal - повертає список рекордів нормального рівня складності, тільки для читання. Так само властивість відзначено атрибутом DataMember, що вказує, що властивість підлягає сериализации;
2. public List <Player> Hard - повертає список рекордів важкого рівня складності, тільки для читання. Так само властивість відзначено атрибутом DataMember, що вказує, що властивість підлягає сериализации;
3. public int CountRecNormal - повертає кількість рекордів нормального рівня складності;
4) public int CountRecHard - повертає кількість рекордів важкого рівня складності.

Так само клас Records має два методи:

1) public void AddRecordNormal (string name, TimeSpan time) – метод додає новий рекорд з ім'ям гравця name і результатом гри time в список рекордів нормального рівня складності, а так само сортує список по зростанню часу проходження рівня;
2) public void AddRecordHard (string name, TimeSpan time) – метод додає новий рекорд з ім'ям гравця name і результатом гри

time в список рекордів важкого рівня складності, а так само сортує список по зростанню часу проходження рівня.

Клас Player має три властивості, зазначених атрибутом DataMember, вказує, що вони підлягають сериализации:

public int Id - повертає унікальний ідентифікатор гравця;

public string Name - повертає ім'я гравця;

public TimeSpan Time - повертає час проходження рівня.

Перерахування Difficulty містить два поля: Normal і Hard, що визначають складність гри нормальну і важку відповідно. Так само відзначені атрибутом DataMember, що вказує, що вони підлягають сериализації. З огляду на те що класи Record і Player є частиною моделі даних, вони повинні бути Серіалізуємі. Так само серіалізуємим має бути і перерахування Difficulty, оскільки його необхідно передавати в HTTP запиті до сервера, а значить, воно буде серіалізовані. А оскільки всі вони розташовані в яку переносять бібліотеці класів, це накладає деякі обмеження на можливості сериализації. Серіалізация для обраних платформ здійснюється з використанням механізму контрактів даних. Тому необхідні для сериализації класи і перерахування повинні бути відзначені відповідним атрибутом DataContract. Клас CalculateRecords призначений для підрахунку часу проходження рівня і виявлення входить результат гри в десятку кращих рекордів в залежності від складності. Містить одне властивість і три методи:

1) public bool IsRunning - повертає true якщо таймер підрахунку часу гри запущений, і false якщо немає;
2) public void Start () - запускає таймер для початку підрахунку часу гри;
3) public bool Stop (Records rec, Difficulty dif, out TimeSpan time) - зупиняє таймер і перевіряє, чи входить результат в десятку кращих результатів rec в залежності від складності dif. Якщо входить, то повертає

значення true, якщо немає - false. У параметр time передається час, витрачений на проходження рівня;

4) public void Reset () - скидає таймер в початковий стан.

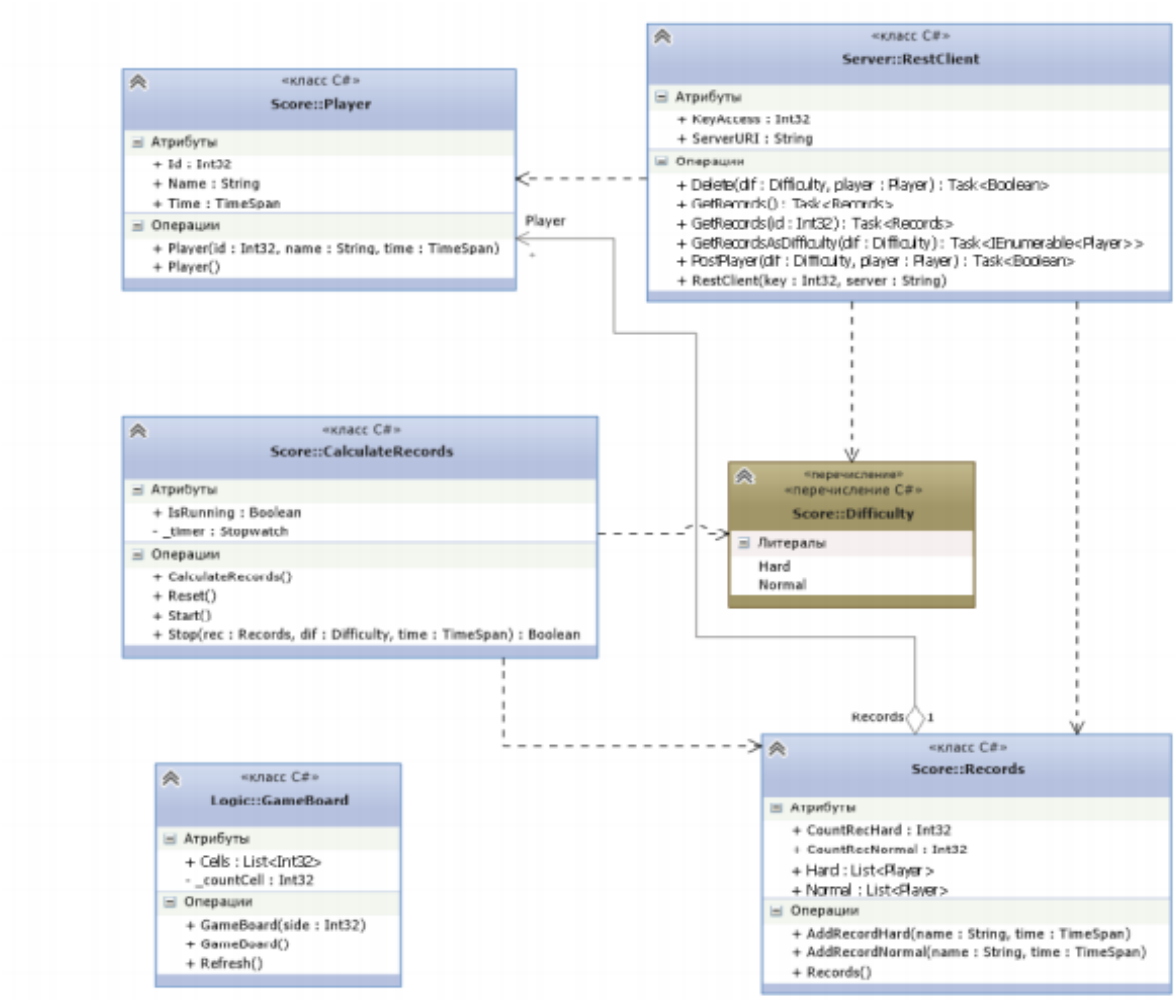
2.1.3 Простір імен Logic

Містить один клас GameBoard, який являє собою ігрове поле. Містить одне властивість і один метод:

1) public List <int> Cells - в кімнаті конференцій осередків поля і код картинки в ній;

2) public void Refresh () - перемішує коди картинок між осередками.

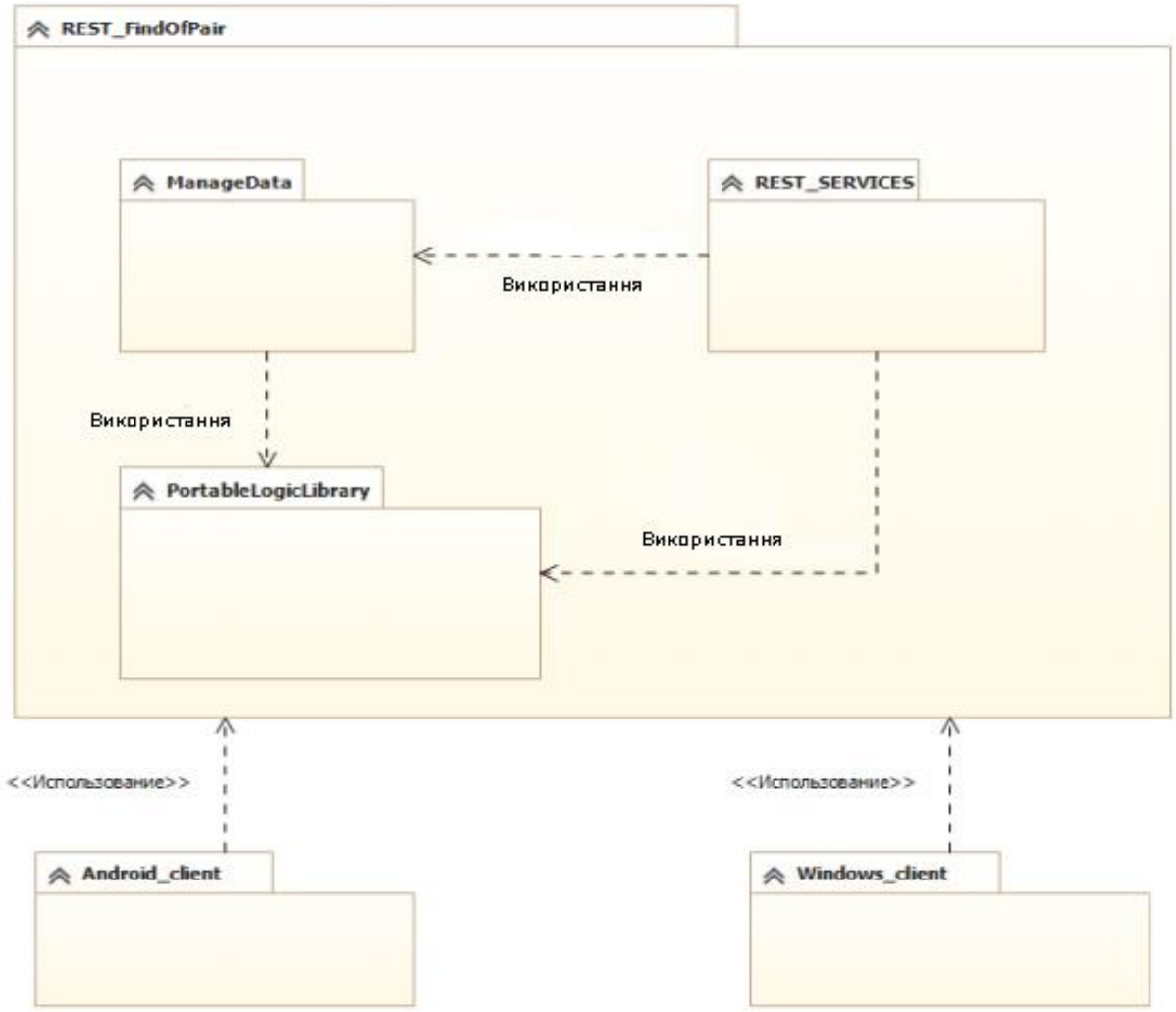
Конструктор GameBoard (int side) приймає як параметр сторону ігрового поля side, і в залежності від неї, створює потрібну кількість осередків і заповнює їх кодами картинок. Максимальний розмір поля 10 на 10 клітин. Якщо вхідний параметр side знаходиться за межами діапазону 1 - 10, то викидається виключення. Конструктор класу без параметрів ініціалізує ігрове поле зі стороною дві клітини. UML діаграма класів для переносної бібліотеки представлена на малюнку 2.



Малюнок 2 - UML діаграма класів PortableLogicLibrary

2.2 Серверна частина

Серверна частина являє собою REST сервіс, який має складну внутрішню архітектуру, включає в себе 3 компонентні частини. UML діаграма пакетів REST сервісу наведена на малюнку 3.



Малюнок 3 - UML діаграма пакетів REST сервісу

Перша частина це модель даних, яка реалізована в переносний бібліотеці класів PortableLogicLibrary.

Друга частина це методи доступу до даних ManageData. Містить інтерфейс IRecordsRepository і клас RecordsRepository.

Інтерфейс IRecordsRepository включає в себе п'ять методів доступу до даними:

1) Records GetAll () - метод повертає всі рекорди;
2) Records Get (int id) - метод повертає всі рекорди гравця з ідентифікатором id;

3) void Post (Player player, Difficulty dif) - метод додає рекорд гравця player в писок рекордів заданої складності dif;

4) bool Delete (Player player, Difficulty dif) метод видаляє рекорд гравця player зі списку рекордів заданої складності dif, і в залежності від

5) успіху / невдачі операції видалення повертає true / false;

6) IEnumerable <Player> GetNormal () - метод повертає список рекордів нормального рівня складності;

7) IEnumerable <Player> GetHard () - метод повертає список рекордів важкого рівня складності.

Клас RecordsRepository успадковує інтерфейс IRecordsRepository і відповідно реалізує всі необхідні методи інтерфейсу. модель даних отримує з бібліотеки PortableLogicLibrary. Для зберігання рекордів на стороні сервера використовується сериализация списку відгуків в форматі JSON. При створенні екземпляра класу з файлу десеріалізуется список рекордів і подальша робота ведеться з ним. При додаванні, видаленні відбувається збереження кожної зміни в серіалізовані файлі.

Третя частина це методи API. Реалізується у вигляді бібліотеки DLL. Основою бібліотеки є клас WorkController, який надає методи API для реалізації CRUD операцій:

1) Records Get (int keyAccess) - отримати всі рекорди з сервера;

2) Records Get (int keyAccess, int id) - отримати всі рекорди з сервера з ідентифікатором гравця id;

3) IEnumerable <Player> Get (int keyAccess, Difficulty dif) – отримати всі рекорди заданої складності dif;

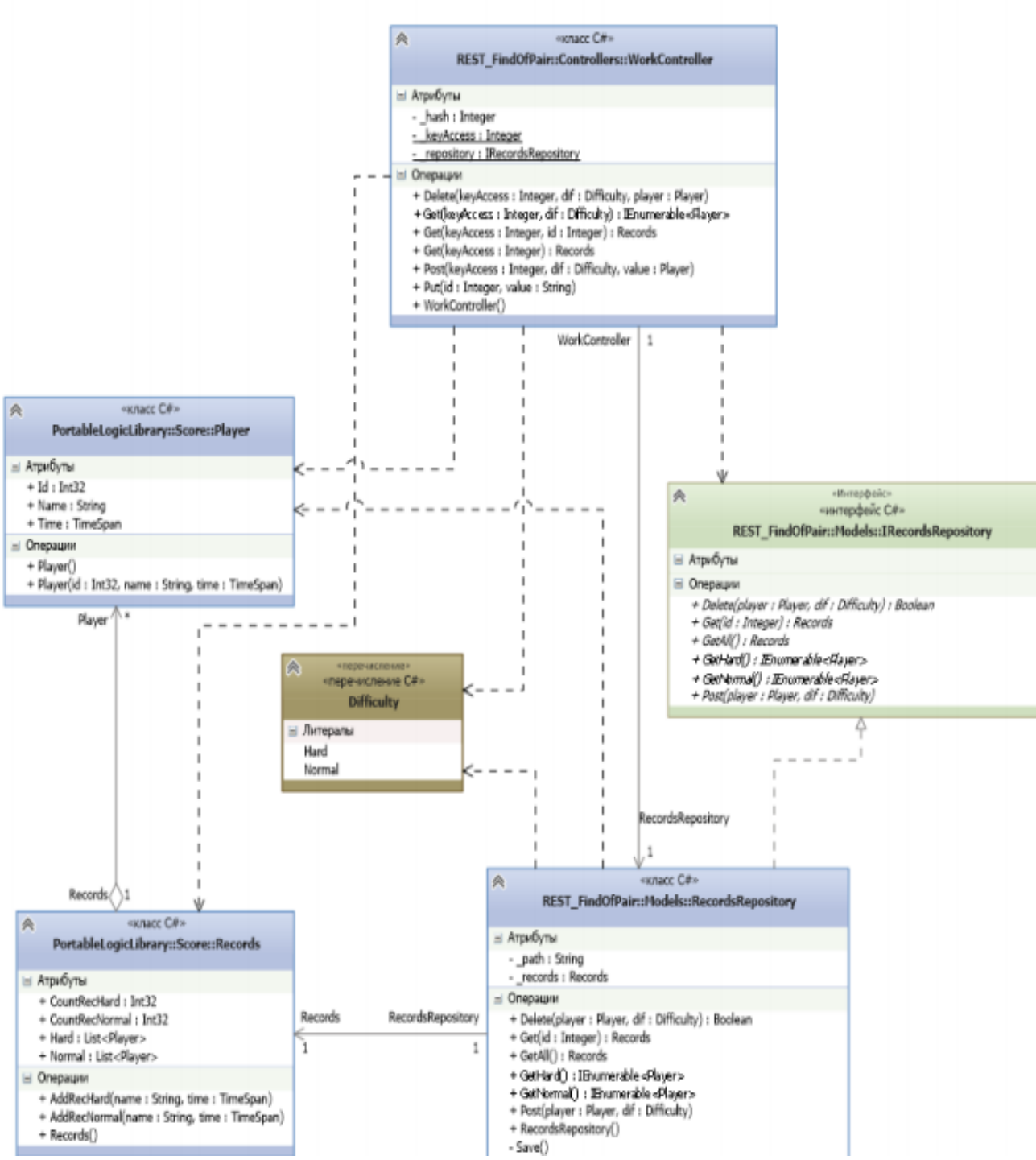
4) void Post (int keyAccess, Player player, Difficulty dif) - додати гравця player в список рекордів заданої складності dif;

5) void Delete (int keyAccess, Player player, Difficulty dif) – видалити рекорд гравця player зі списку рекордів заданої складності dif.

Окремо слід відзначити призначення параметра keyAccess в кожному методі API. Оскільки серверна частина реалізована у вигляді REST сервісу, то доступ до його API можливий не тільки з ігрових клієнтів, але і з браузерів і інших мережевих програм. Тим самим потенційно виникає вразливість сервісу до несанкціонованого доступу до даних, наприклад, склавши HTTP запит із заданими параметрами можна відіслати на сервер свідомо помилковий результат або видалити існуючий. Спосіб авторизації через реєстрацію акаунтів теж не підходить, оскільки точно так же можна з браузера зареєструвати фіктивний аккаунт і з його допомогою отримати доступ до даних. Тому рішенням є введення додаткового параметра keyAccess, який містить особливу комбінацію цифр, що є ключем доступу, відомий тільки розробнику клієнтських додатків і сервера. Цей ключ зашитий в виконуваний код, і зловмиснику витягти його буде досить проблематично, як і просто підібрати, з огляду на його складності - 10 знаків. Для того щоб зловмисник не перехопив ключ доступу при його передачі серверу, ключ доступу хешіруется і при відправці запиту до сервера клієнт передає хеш ключа доступу, який порівнюється з хешем на стороні сервера і якщо вони збігаються, то сервер виконує запитану операцію.

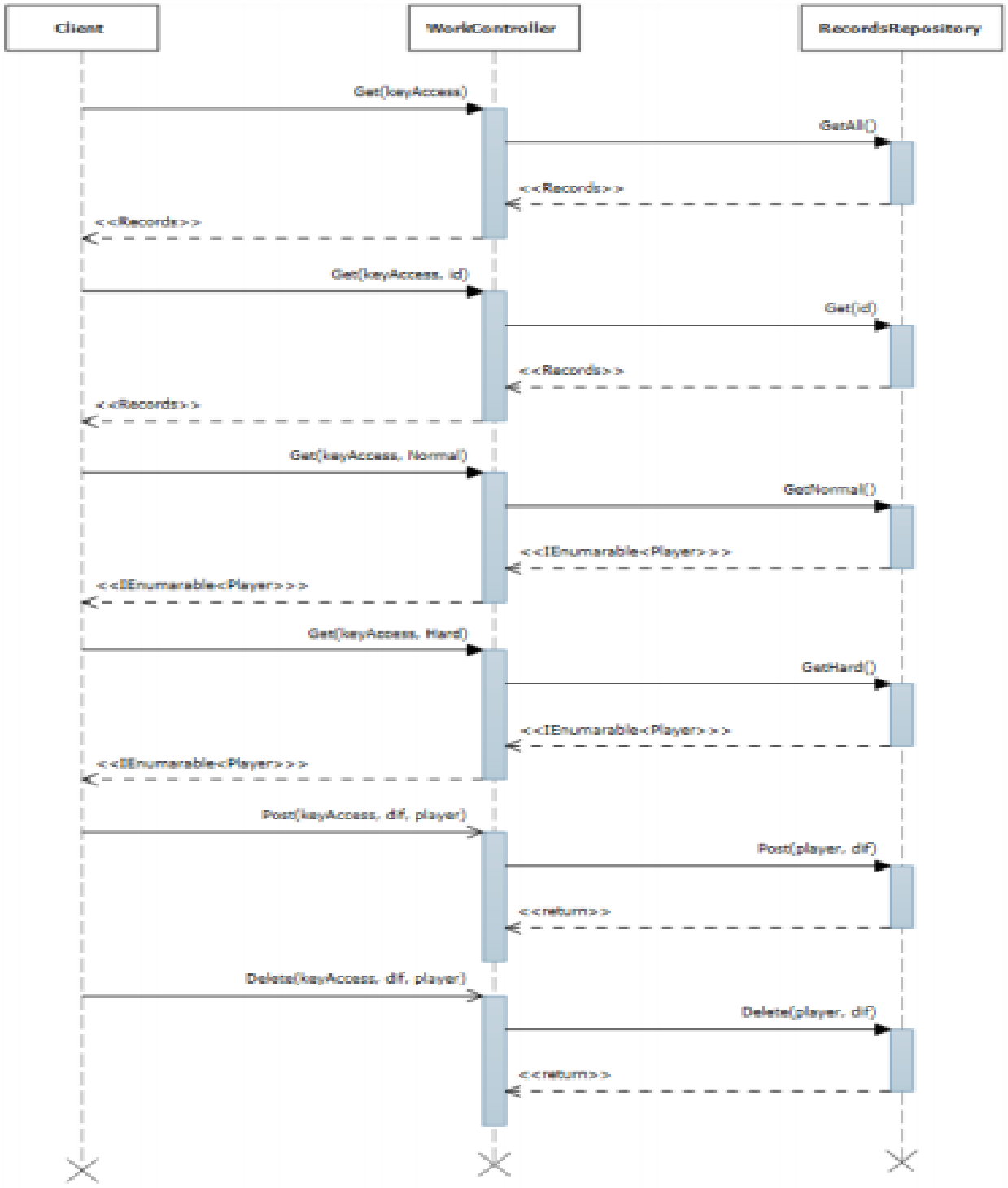
Через інтерфейс IRecordsRepository клас WorkController отримує доступ до даних.

UML діаграма класів REST_FindOfPairs приведена на малюнку 4.



Малюнок 4 - UML діаграма класів REST_FindOfPairs

UML діаграма послідовностей REST_FindOfPairs приведена на малюнку5.



Малюнок 5 - UML діаграма послідовностей REST_FindOfPairs

При проектуванні додатків для Android необхідно враховувати деякі особливості платформи і життєвого циклу програми. Екран додатки представляє собою в'язку класу Activity і інтерфейсу на мові XAML. Тим самим інтерфейс відділений від логіки програми, але не має в собі ніяких функцій, а

зберігає лише розмітку екрана і опис елементів управління на ньому. Для кожного екрану при ініціалізації необхідно вказати шар з розміткою, яку буде мати відмалювати екран. В процесі роботи при необхідності зміни екрану можна підміняти цей шар, тим самим змінювати розмітку на екрані. У цьому випадку необхідна всього один екземпляр класу Activity, але це не найкраще рішення в проектуванні додатки в цілому, тому що код втрачає структурованість, стає більш заплутаним і громіздким. Однак для виняткових ситуацій це буде розумним способом зміни екрану.

Кращим рішенням є створення для кожного екрану свій шар розмітки і свій екземпляр класу Activity, який містить необхідний функціонал для даного екрану. Всього в додатку передбачено сім екранів, відповідно сім розміток XAML:

1) головний екран - містить кнопки навігації: грати, настройки, локальні рекорди, глобальні рекорди, правила гри;
2) ігровий екран нормального рівня складності - містить ігрове поле у вигляді таблиці 4 на 4 клітини (клітина представлена елементом управління ImageView), нижче розташовані дві кнопки: заново і назад;
3) ігровий екран важкого рівня складності - містить ігрове поле у вигляді таблиці 6 на 6 клітин (клітина представлена елементом управління ImageView), нижче розташовані дві кнопки: заново і назад;
4) екран локальних рекордів - містить дві кнопки: нормальний і складний, кожна з яких відображає відповідну таблицю з рекордами для пристрою;
5) екран світових рекордів - містить дві кнопки: нормальний і складний, кожна з яких відображає відповідну таблицю з глобальними рекордами;
6) екран налаштувань - містить елемент керування RadioGroup, що дозволяє вибрати складність гри;
7) екран довідки - відображає правила гри.

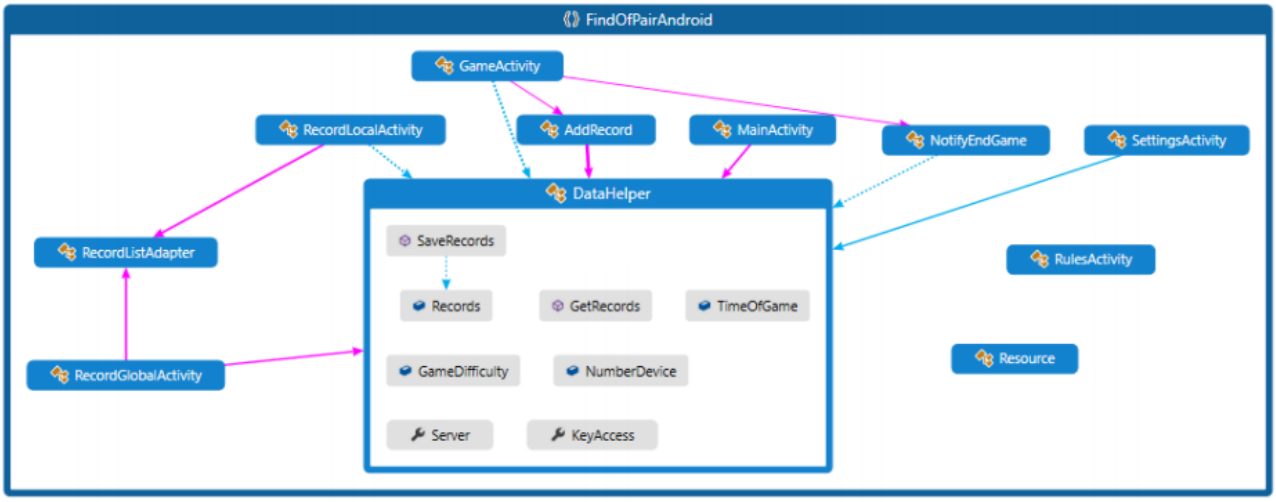
Але класів з логікою екранів всього шість, кожен з них успадковується від базового класу Activity:

1) MainActivity - реалізує логіку роботи головного екрану програми;
2) GameActivity - реалізує логіку роботи безпосередньо ігрового процесу для обох рівнів складності;
3) RecordGlobalActivity - реалізує логіку роботи екрану світових рекордів;
4) RecordLocalActivity - реалізує логіку роботи екрану локальних рекордів;
5) RulesActivity - реалізує логіку роботи екрану з правилами гри;
6) SettingsActivity - реалізує логіку роботи екрану налаштувань.

Так само для реалізації всіх функцій необхідно чотири класи:

1) DataHelper - допоміжний клас який забезпечує передачу даних між екземплярами екранів і містить ряд методів для обробки і підготовки до подання даних;
2) AddRecord - клас, забезпечує реалізацію вікна повідомлення про новий рекорд з можливістю збереження на сервері;
3) NotifyEndGame - клас, забезпечує реалізацію вікна повідомлення про закінчення гри, коли рекорд не був встановлений;
4) RecordListAdapter - клас, забезпечує реалізацію адаптера, який створює особливі елементи управління для відображення рядка з результатом гри в заданому вигляді та організовує їх у вигляді списку.

Карта коду програми для Android, яка відображає взаємодію між класами, представлена на малюнку 6.



Малюнок 6 - Карта коду програми для Android

На карті коду прийняті наступні позначення ліній:

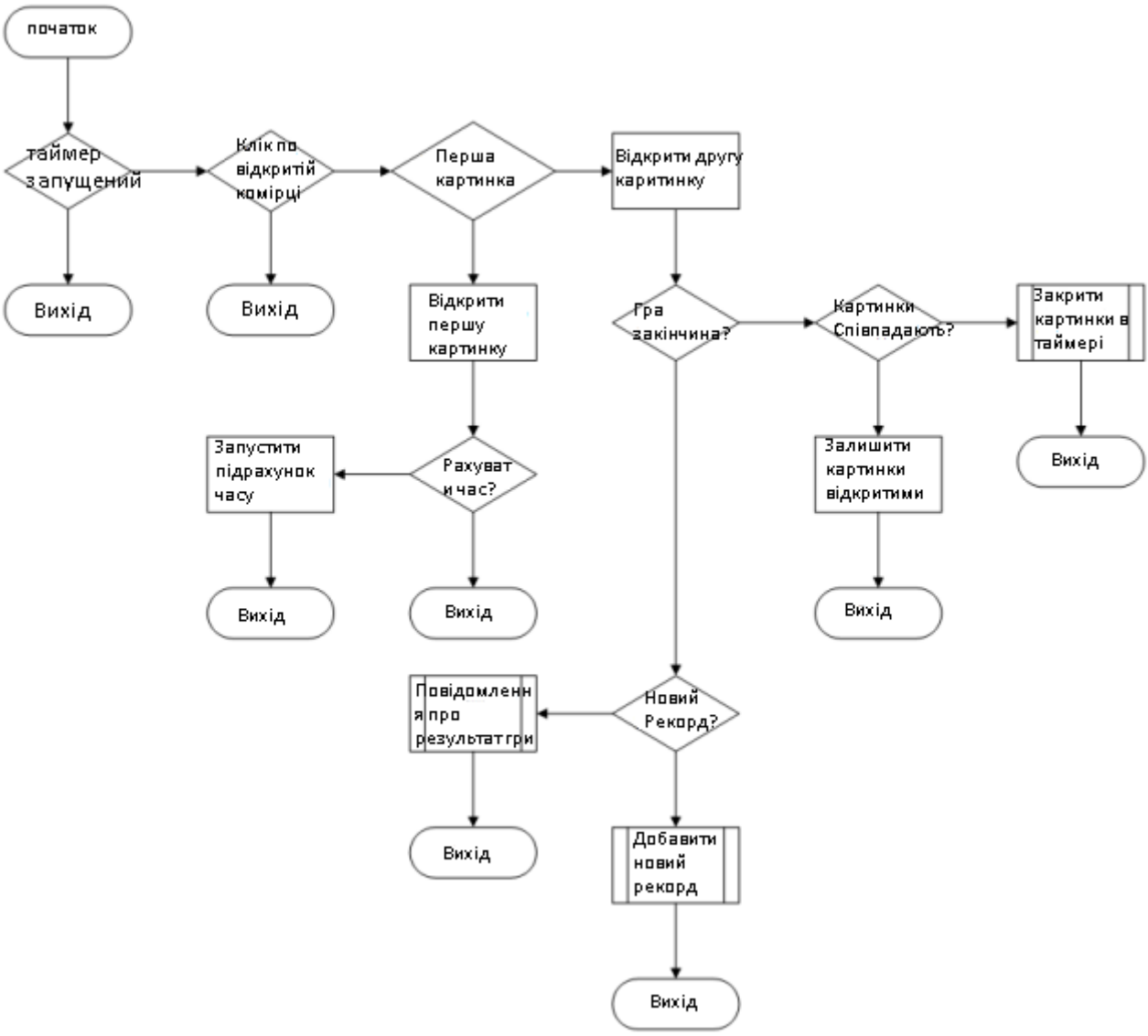
1) рожева суцільна лінія - виклики методів класу;
2) блакитна суцільна лінія - запис поля класу;
3) блакитна пунктирна лінія - читання поля класу.

Клас GameActivity. При ініціалізації в залежності від обраної складності гри встановлює відповідну розмітку екрана і ініціалізує модель ігрового поля заданого розміру. Тут використовується схема:

один керуючий клас - дві розмітки. Це обумовлено тим, що відмінності в логіці мінімальні і обмежуються лише різницею в розмірі ігрового поля. Включає в себе ряд методів:

1) private void AssignImageNormal () - метод зіставляє картинки між елементами управління, з яких складається ігрове поле при нормальному рівні складності;
2) private void AssignImageHard () - метод зіставляє картинки між е лементами управління, з яких складається ігрове поле при нормальному рівні складності;
3) private bool CheckWin () - метод повертає true, якщо всі клітини ігрового поля відкриті, і false, якщо немає;

4) private void GameActivity_Click (object sender, EventArgs e) – основний обробник, реалізує саме логіку гри з використанням моделі даних і моделі ігрового поля з переносною бібліотеки класів. Алгоритм методу представлений на малюнку 7.



Малюнок 7 - Алгоритм обробника кліка по клітці

Використання таймера затримки приховування пари різних картинок має деякі особливості. Оброблювач таймера приховує картинки, шляхом підміни зображень на елементі управління, що представляє окрему клітинку. Доступ до елементів управління (інтерфейсу користувача в цілому) є тільки у головного потоку, а обробник таймера виконується в окремому потоці і не може оновити зображення на елементах управління. Для обходу цього обмеження використовується спеціальний об'єкт Handler, який надає доступ до черги повідомлень на виконання в головному потоці, шляхом отримання повідомлень з інших потоків, і при необхідності додає їх в чергу повідомлень (дій) для виконання в головному потоці. Тому в обробнику таймера до об'єкта Handler відправляється спеціальне повідомлення, а вже в обробнику цієї об'єкта (який виконується в головному потоці) виконується приховування пари різних

картинок. За результатом завершення гри викликається діалогове вікно повідомлення про результати гри, на якому відображається час проходження рівня і, якщо час входить в 10 кращих для пристрою, поле для введення імені гравця і кнопка для збереження рекорду. Статичний допоміжний клас DataHelper. Включає в себе шість

полів і два методи:

1) public static Difficulty GameDifficulty - поле зберігає поточну складність гри;
2) public static TimeSpan TimeOfGame - поле зберігає результат останньої гри;
3) public static Records Records - зберігає таблицю рекордів;
4) public static int KeyAccess - зберігає ключ доступу до сервера;
5) public static string Server - зберігає адресу сервера;
6) public static int NumberDevice - зберігає ідентифікатор пристрою, який представляє id користувача для ідентифікації його в глобальному сховище рекордів серед інших гравців;
7) public static Records GetRecords () - метод здійснює читання локальної таблиці рекордів на пристрої;
8) public static void SaveRecords (Records record) - метод здійснює збереження локальної таблиці рекордів на пристрої.

2.3 Клієнтська частина на платформі Android

Розробка версії гри для Windows є більш простим завданням, оскільки не має тих обмежень, які мали місце при розробці під Android. Наприклад, обробник подій таймера так само виконується в окремому потоці, але все одно має доступ до елементів управління інтерфейсу користувача, а значить приховувати картинки можна безпосередньо з коду його обробника. Так само не потрібно спеціального допоміжного класу DataHelper для організації передачі даних, оскільки всі необхідні екземпляри екранів знаходяться в занедбаному стані, і передача потрібних даних можлива простим зверненням до членів класу використовуючи посилання на необхідний клас.

Інтерфейс реалізується з використанням WindowsForm, потрібно всього шість вікон:

1) головне вікно (клас MainWindow) - містить ігрове поле, яке змінюється в залежності від обраної складності гри. Ігрове поле являє собою контейнер з елементами управління Image. У верхній частині

розташовано меню з кнопками: меню, складність, таблиця рекордів, довідка;

2) правила гри (клас Rules) - містить інформацію про правила гри;

3) локальна таблиця рекордів (клас TableOfLocalRecords) - містить таблицю локальних рекордів;

4) глобальна таблиця рекордів (клас TableOfGlobalRecords) - містить таблицю світових рекордів;

Використання таймера затримки приховування пари різних картинок має деякі особливості. Оброблювач таймера приховує картинки, шляхом підміни зображень на елементі управління, що представляє окрему клітинку. Доступ до елементів управління (інтерфейсу користувача в цілому) є тільки у головного потоку, а обробник таймера виконується в окремому потоці і не може оновити зображення на елементах управління. Для обходу цього обмеження використовується спеціальний об'єкт Handler, який надає доступ до черги повідомлень на виконання в головному потоці, шляхом отримання повідомлень з інших потоків, і при необхідності додає їх в чергу повідомлень (дій) для виконання в головному потоці. Тому в обробнику таймера до об'єкта Handler відправляється спеціальне повідомлення, а вже в обробнику цієї об'єкта (який виконується в головному потоці) виконується приховування пари різних картинок.

За результатом завершення гри викликається діалогове вікно повідомлення про результати гри, на якому відображається час проходження рівня і, якщо час входить в 10 кращих для пристрою, поле для введення імені гравця і кнопка для збереження рекорду. Статичний допоміжний клас DataHelper. Включає в себе шість полів і два методи:

1. public static Difficulty GameDifficulty - поле зберігає поточну
2. складність гри;
3. public static TimeSpan TimeOfGame - поле зберігає результат останньої гри;
4. public static Records Records - зберігає таблицю рекордів;
5. public static int KeyAccess - зберігає ключ доступу до сервера;
6. public static string Server - зберігає адресу сервера;
7. public static int NumberDevice - зберігає ідентифікатор пристрою, який представляє id користувача для ідентифікації його в глобальному сховище рекордів серед інших гравців;
8. public static Records GetRecords () - метод здійснює читання локальної таблиці рекордів на пристрої;

9. public static void SaveRecords (Records record) - метод здійснює збереження локальної таблиці рекордів на пристрої.

2.4 Клієнтська частина на платформі Windows

Розробка версії гри для Windows є більш простим завданням, оскільки не має тих обмежень, які мали місце при розробці під Android. Наприклад, обробник подій таймера так само виконується в окремому потоці, але все одно має доступ до елементів управління інтерфейсу користувача, а значить приховувати картинки можна безпосередньо з коду його обробника. Так само не потрібно спеціального допоміжного класу DataHelper для організації передачі даних, оскільки всі необхідні екземпляри екранів знаходяться в занедбаному стані, і передача потрібних даних можлива простим зверненням до членів класу використовуючи посилання на необхідний клас.

Інтерфейс реалізується з використанням WindowsForm, потрібно всього шість вікон:

1. головне вікно (клас MainWindow) - містить ігрове поле, яке змінюється в залежності від обраної складності гри. Ігрове поле являє собою контейнер з елементами управління Image. У верхній частині розташовано меню з кнопками: меню, складність, таблиця рекордів, довідка;
2. правила гри (клас Rules) - містить інформацію про правила гри;
3. локальна таблиця рекордів (клас TableOfLocalRecords) - містить таблицю локальних рекордів;
4. глобальна таблиця рекордів (клас TableOfGlobalRecords) - містить таблицю світових рекордів;

5) додати новий рекорд (клас AddRecord) - являє вікнодля збереження нового рекорду як локально, так і глобально;

6) про гру (клас AboutGame) - містить інформацію про гру.

Алгоритм обробника кліка по клітці такий же, як у версії для Android. У глобальній таблиці рекордів встановлений рекорд на цьому комп'ютері виділяється кольором, щоб його можна було відрізнити від інших. Локальні рекорди зберігаються серіалізованом вигляді в каталозі розгортання додатки.

2.5 Результат розробки

В результаті розробки була спроектована переносна бібліотека класів, що є основою програми, «Знайди пари» являє собою реалізацію гри типу Memories.

Ігрове поле являє собою 16 і 64 клітини, що містять приховані пари картинок. Мета гри - відкрити всі картинки. Картинки відкриваються парами, спочатку перша, потім друга, якщо картинки однакові, то вони обидві залишаються відкритими і так до тих пір, поки все полі не буде відкрито.

Для початку гри запустіть додаток. Відкриється головне вікно як на рисунку 2.5.1. У верхній частині вікна розташовані чотири області: «Меню»,«Складність», «Таблиця рекордів» та «Довідка».

Малюнок 2.5.1- Головне вікно програми.

У грі передбачено два рівня складності:

- нормальний - ігрове поле складається з 16 клітин;
- важкий - ігрове поле складається з 36 клітин.

За замовчуванням рівень складності гри встановлений як «Нормальний». Змінити складність можна у відповідному меню. Ігровий процес відображений на малюнку 2.5.2

Малюнок 2.5.2- ігровий процес

По закінченню гри ігрове поле буде заповнено картинками. Для виборі рівня складності потрібно перейти в відповідний пункт меню і вибрати з двох існуючих. Після закінчення гри у разі якщо гравець встановив рекорд, йому пропонується ввести своє ім'я для збереження результату в таблиці рекордів. Також є можливість переглянути попередні рекорди інших гравців для цього потрібно вибрати відповідний пункт меню .

3 ФІНАНСОВИЙ МЕНЕДЖМЕНТ

3.1 Введення

В даний час перспективність наукового дослідження визначається не стільки масштабом відкриття, оцінити яке на перших етапах життєвого циклу високотехнологічного і ресурсоефективного продукту буває досить важко, скільки комерційною цінністю розробки. Оцінка комерційної цінності розробки є необхідною умовою при пошуку джерел фінансування для проведення наукового дослідження і комерціалізації його результатів. Особливо це важливо для розробників комерційного програмного забезпечення, оскільки конкуренція на ринку мобільних і десктопних додатків дуже висока.

Метою даного розділу є проектування конкурентоспроможною гри «Знайди пари», розробленої з використанням технології XAMARIN.

Завданнями даного розділу є оцінка комерційного потенціалу і перспективності проведення розробки, а так само визначення сильних і поліпшення слабких сторін розробляється.

Процес вирішення цих завдань буде складатися з аналізу існуючих конкурентних технічних рішень і SWOT аналіз програми.

3.2 Аналіз конкурентних технічних рішень

Детальний аналіз конкуруючих розробок, існуючих на ринку, необхідно проводити систематично, оскільки ринки перебувають в постійному русі. Такий аналіз допомагає вносити корективи в процес розробки, щоб успішніше протистояти своїм суперникам. Важливо реалістично оцінити сильні і слабкі сторони розробок конкурентів.

З цією метою може бути використана вся наявна інформація про конкурентних розробках:

- технічні характеристики розробки;
- конкурентоспроможність розробки;
- рівень завершеності наукового дослідження (наявність макета, прототипу і т.п.);
- бюджет розробки;
- рівень проникнення на ринок.

Аналіз конкурентних технічних рішень з позиції ресурсоефективності та ресурсозбереження дозволяє провести оцінку порівняльної ефективності наукової розробки і визначити напрямки для її майбутнього підвищення.

Даний аналіз проводиться за допомогою оціночної карти конкурентів.

В даний момент у розробки є кілька конкурентів: мобільне додаток для Android «Знайди дублі», мобільний додаток для Windows 10 Mobile «Тренування пам'яті» і браузерна гра «Memories», їх аналіз представлений в таблиці 1.

Критерії оцінки	Вага критерію	Оцінки				Конкурентоспроможність			
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$Б_{к3}$	$К_ф$	$К_{к1}$	$К_{к2}$	$К_{к3}$
1. Збільшення продуктивності праці	0.02	4	4	5	3	0.08	0.08	0.1	0.06
2. Зручність використання	0.14	5	4	5	2	0.7	0.56	0.7	0.28
3. Стресостійкість	0.02	5	3	4	2	0.1	0.06	0.08	0.04
4. Енергоекономність	0.08	5	4	3	3	0.4	0.32	0.24	0.24
5. Надійність	0.03	5	3	4	2	0.15	0.09	0.12	0.06
6. Рівень шуму	0.01	5	5	5	5	0.05	0.05	0.05	0.05
7. Безпека	0.01	5	5	5	2	0.05	0.05	0.05	0.02
8. Потреба в ресурсах пам'яті	0.01	5	4	4	3	0.05	0.04	0.04	0.03
9. функціональна потужність	0.1	5	2	4	2	0.5	0.2	0.4	0.2
10. простота експлуатації	0.07	5	4	4	3	0.35	0.28	0.28	0.21
11. якість інтерфейсу	0.14	4	5	4	2	0.56	0.7	0.7	0.28
12. можливості підключення в мережу ЕОМ	0.07	5	1	3	5	0.35	0.07	0.07	0.35

Критерії оцінки	Вага крит ерію	Оцінки				конкурентоспроможність			
		$Б_ф$	$Б_{к1}$	$Б_{к2}$	$Б_{к3}$	$К_ф$	$К_{к1}$	$К_{к2}$	$К_{к3}$
Економічний критерій оцінки ефективності									
1. конкурентоспроможність товару	0.1	4	3	2	1	0.4	03	0.2	0.1
2. Рівень проникнення на ринок	0.1	1	2	1	4	0.1	0.2	0.1	0.4
3. Ціна	0.03	5	5	1	5	0.15	0.15	0.03	0.15
4. Передбачуваний термін експлуатації	0.01	3	3	2	3	0.03	0.03	0.02	0.03
5. Післяпродажне обслуговування	0.01	3	1	1	4	0.03	0.01	0.01	0.04
6. Фінансування наукової розробки	0.02	1	1	1	2	0.02	0.02	0.02	0.04
7. Термін виходу на ринок	0.02	3	5	2	4	0.06	0.1	0.04	0.08
8. Наявність сертифікаційної розробки	0.01	1	1	1	1	0.01	0.01	0.01	0.01
Сума	1	79	65	61	58	4.14	3.32	3.26	2.67

Примітка:
1) Ф - дана розробка;
2) к1 - мобільний додаток для Android «Знайди дублі»;
3) к2 - мобільний додаток для Windows 10 Mobile «Тренування пам'яті»;
4) к3 - браузерна гра «Memories».

На основі даної оціночної карти можна зробити висновок, що дана розробка перевершує своїх конкурентів за більшістю параметрів і в цілому є кращим рішенням на серед них.

Уразливість конкурентів обумовлена декількома позиціями.

Для гри «Знайди дублі» характерна низька стійкість, обмежений функціонал, неможливість підключення до ЕОМ, слабке проникнення на ринок.

Для гри «Тренування пам'яті» характерна незбалансоване енергоспоживання, обмежені можливості роботи з мережею, рівень проникнення на ринок найнижчий, немає безкоштовної версії.

Для гри «Memories» характерна низька надійність, ергономіка та зручність для роботи користувача залишають бажати кращого, низький рівень безпеки і низька функціональність.

Розроблюване кроссплатформне додаток «Знайди пари» спроектовано з тим розрахунком, щоб всі ці недоліки усунути і зіграти на слабких сторонах конкурентів, тим самим різко підвищивши конкурентоспроможність додатки. Так само не обійшлися без розгляду і сильні боку конкурентів, які розробляється програма реалізує приблизно на тому ж рівні що і конкуренти. Але, тим не менше, загрози з боку конкурентів існують, тому необхідно розглянути шляхи розвитку додатки на основі сильних і слабких його сторін. Для цього використовується методика розгляду сильних і слабких сторін, а так же можливостей і загроз із застосуванням SWOT аналізу.

3.3 SWOT аналіз

SWOT - Strengths (сильні сторони), Weaknesses (слабкі сторони), Opportunities (можливості) і Threats (загрози) - являє собою комплексний аналіз розроблюваного проекту. SWOT аналіз застосовують для дослідження зовнішнього і внутрішнього середовища проекту. [14]

Матриця SWOT аналізу представлена в таблиці 2.

	Сильні сторони 1)Кросплатформеність 2)Підтримка сучасних платформ 3)Клієнт-серверна архітектура 4)Отказоустойчивость клієнтів 5)Постійна технічна підтримка 6)Висока продуктивність 7)Низькі системні вимоги	Слабкі сторони 1) Один сервер 2) Мала розкрутка серед користувачів 3) Один набір картинок для ігрових полів
Можливості 1)Перенесення додатка на інші платформи 2)Розгортання серверів на декількох комп'ютерах 3)Рекламна кампанія в соцмережах, журналах,	1)Розширити кількість підтримуваних платформ (iOS, WinPhone, Windows 10 Mobile) 2)Оптимізація роботи мережі 3)Доопрацювання інтерфейсу	1)Активна рекламна кампанія серед потенційних користувачів 2)У міру збільшення кількості гравців збільшувати кількість серверів

пошукових системах 4)Додавання нових рівнів в гру 5)Поліпшення інтерфейсу		3)Розширити набір картинок для ігрових полів 4)Розширення доступних рівнів
Загрози 1) Втрата гравців через відмови одного сервера 2) Малий коефіцієнт конвертації кількості показів реклами в гравців 3) Відтік гравців в зв'язку втратою інтересу до неї 4) Розвиток конкурентів	1) Розгорнути резервний сервер 2) Розвиток ігрового процесу 3) Збільшення темпів реалізації планів на зростання і розвиток додатки	1) Збільшення стабільності роботи серверної частини 2) Аналіз причин розвитку конкурентів і перейняття їхнього досвіду

3.4 Висновок розділу

В даному розділі проведений аналіз конкурентів і SWOT аналіз, поставлена мета досягнута, завдання вирішені. Розглянуто сильні і слабкі боку конкурентів, також виявлені свої сильні і слабкі сторони. Определѐн шлях розвитку в майбутньому. Можна сказати, що було спроектовано конкурентоспроможне додаток на базі нової сучасної технології, яка готова до виходу на ринок.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Враховуючи, що при використання автоматичної системи управління мінімаркетом, користувачі проводять більшу частину свого робочого часу за комп’ютером, необхідно дотримуватися правил охорони праці при роботі з персональним комп’ютером.

Питання охорони праці регулюються певними законодавчими та нормативно-правовими актами, які, зокрема, визначають обов’язки роботодавця із забезпечення робітникам комфортних та безпечних умов для здійснення роботи. Ці обов’язки, а також права робітників на таких умовах праці передбачені частиною 2 ст.2 і частиною 1 ст.21 КЗпП, а також ст.13

Закону України «Про охорону праці», у яких визначаються основні положення з реалізації конституційного права робітників.

Існує цілий ряд вимог, які визначають специфіку заходів з охорони праці при роботі з персональним комп'ютером. Законодавчі та нормативноправові акти, які за участі відповідних органів державної влади регулюють відносини між роботодавцем та робітником з питань безпеки, гігієни праці та виробничого середовища, а також встановлюють єдиний порядок організації охорони праці в Україні. На їх основі розроблені чисельні документи: правила, інструкції, норми, державні санітарні правила та ін., якими мають керуватись роботодавці та які регламентують певні питання щодо конструкції електронно-обчислювальної техніки, та особливостей їх розміщення [17].

На сьогодні основними документами, які регламентують питання охорони праці при використанні працівниками персональних комп'ютерів, можна вважати наступні підзаконні акти:

1) НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями»;

2) ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи 80 з візуальними дисплейними терміналами електронно-обчислювальних машин»;

У відповідності з цими актами, при розробці будь-якого програмного забезпечення, в тому числі при розробці автоматизованої системи управління мінімаркетом, необхідно вжити всіх необхідних заходів з охорони праці та розробити відповідні документи, зокрема, але не виключно:

• положення про охорону праці; • інструкції з охорони праці для касира та адміністратора;

• накази з охорони праці;

• журнали інструктажу, реєстрацій та інше. Окрім цього, в залежності від кількості працівників в ІТ компанії та від кількості працівників задіяних в проекті по розробці системи управління мінімаркетом служба охорони праці виглядає наступним чином:

• В ІТ компанії або проекті з кількістю працюючих 50 і більше осіб, роботодавець створює службу охорони праці відповідно до типового положення, що затверджується центральним органом виконавчої влади, що забезпечує формування державної політики у сфері охорони праці.

- В ІТ компанії або проекті з кількістю працюючих менше 50 осіб, функції служби охорони праці можуть виконувати в порядку сумісництва особи, які мають відповідну підготовку.

- В ІТ компанії або проекті з кількістю працюючих менше 20 осіб, для виконання функцій служби охорони праці можуть залучатися сторонні спеціалісти на договірних засадах, які мають відповідну підготовку.

Підпорядковується служба охорони праці згідно із законодавством безпосередньо роботодавцеві [17].

Проте роботодавець може доручити функціональне управління (кураторство) діяльністю служби іншій посадовій особі, скажімо, головному інженерові, заступникові директора з охорони праці тощо.

Покладення таких обов'язків потрібно закріпити наказом або відобразити в посадовій інструкції уповноваженої особи.

Робота служби охорони праці підприємства має здійснюватися відповідно до плану роботи та графіків обстежень, затверджених роботодавцем.

Функції служби охорони праці:

1. Підготовка проектів наказів (розпоряджень) з питань охорони праці і внесення їх на розгляд роботодавцю. Проведення спільно з представниками інших структурних підрозділів і за участю представників професійної спілки підприємства або, за її відсутності, уповноважених найманими працівниками осіб із питань охорони праці перевірок дотримання працівниками вимог нормативно-правових актів з охорони праці.

2. Проведення з працівниками вступного інструктажу з питань охорони праці та супутніх інструктажів.

3. Ведення обліку та проведення аналізу причин виробничого травматизму, професійних захворювань, аварій на виробництві, заподіяної ними шкоди.

4. Забезпечення належного оформлення і зберігання документації з питань охорони праці, а також своєчасної передачі її до архіву для тривалого зберігання згідно з установленим порядком.

5. Складання звітності з охорони праці за встановленими формами.

6. Складання за участю керівників підрозділів підприємства переліків професій, посад і видів робіт, на які повинні бути розроблені інструкції з охорони праці, що діють в межах підприємства, надання методичної допомоги під час їх розроблення.

7. Інформування працівників про основні вимоги законів, інших нормативно-правових актів та актів з охорони праці, що діють в межах підприємства.

8. Розгляд питань про підтвердження наявності небезпечної виробничої ситуації, що стала причиною відмови працівника від виконання дорученої роботи відповідно до законодавства (у разі необхідності).

9. Організація - забезпечення підрозділів нормативно-правовими актами з охорони праці та актами з охорони праці, що діють в межах підприємства, посібниками, навчальними матеріалами з цих питань [17].

Дотримання даних пунктів є необхідним при роботі з автоматизованою системою управління мінімаркетом.

4.2 Безпека в надзвичайних ситуаціях

Загальні ергономічні вимоги для організації робочого місця користувача ПЕОМ (ГОСТ 12.2.049-80, ГОСТ 122032-78, ГОСТ 22269-76). Ці вимоги встановлюють основні параметри робочого місця, оснащеного дисплеєм, і враховують особливість виконуваних робіт.

4.2.1 Параметри робочого місця

Площа кабінету, в якому буде проходити робота повинна бути не менш 6 м², а об’єм не менш 24 м³. Для внутрішньої обробки приміщення повинні використовуватися дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі – 0,7-0,8; для стін – 0,5-0,6; для підлоги – 0,3-0,5. Конструкція робочого столу повинна забезпечувати оптимальне розміщення на робочій поверхні використовуваного обладнання.

Конструкція крісла повинна забезпечувати підтримку раціональної робочої пози під час роботи з відео-дисплейним терміналом (Далі ВДТ) і ПЕОМ, дозволяти змінювати позу з метою зниження статичного напруження м'язів шийно-плечової області і спини для попередження розвитку втоми працюючого (згідно з ГОСТ 12.2.032-78). Поверхня сидіння, спинки та інших елементів стільця (крісла) повинна бути напівм'якою, з покриттям, що не

електризується, неслизьке та повітронепроникне, що забезпечує легке очищення від забруднення.

Висота робочої поверхні столу, за відсутності можливості її регулювання повинна складати 725 мм. Робочий стіл повинен мати простір для ніг висотою не менше 600 мм, шириною – не менше 500 мм, не менше 450 мм в глибину на рівні колін і на рівні простягнутої ноги – не менше 650 83 мм. Робоче місце має бути обладнане підставкою для ніг, має ширину не менше 300 мм, глибину не менше 400 мм, регулювання по висоті в межах 150 мм за кутом нахилу опорної поверхні підставки до 20 градусів.

Відстань від очей користувача до екрану дисплея має становити 500- 700 мм.

Кут зору 10-20°, але не більше 40°; кут між верхнім краєм дисплея і рівнем очей користувача має становити не менше 10°. Кращим є розташування екрану перпендикулярно до лінії зору користувача. Робочі місця по відношенню до світлових прорізів повинні розташовуватися не ближче 3 м так, щоб природне світло падало збоку, переважно зліва. Освітленість також впливає на стан здоров'я і працездатність людини. У відповідності зі СНіП 11-4-79 встановлені наступні вимоги до освітленості: Для штучного освітлення:

- Комбіноване освітлення – освітленість 1500 лк;
- Загальне освітлення – освітленість 400 лк. Для природного освітлення:
- Верхнє або комбіноване освітлення – коефіцієнт природної освітленості (далі КПО) 10%;
- Бічне освітлення – КПО 3.5%. Для суміщеного освітлення:
- Верхнє або комбіноване освітлення – КПО 3-6%;
- Бічне освітлення – КПО 1.1-2%.

До основних показників, що визначають умови здорової роботи, належать: фон, контраст об'єкта з фоном, видимість, показник осліпленості, коефіцієнт пульсації освітленості.

Фон характеризується коефіцієнтом відбиття. Контраст об'єкта з фоном (К) характеризується співвідношенням яскравості розглянутого об'єкта (точки, лінії, знаки) і фону. Оскільки роботи користувача ПЕОМ відносяться до категорії 1а – легкі фізичні роботи (роботи проводяться сидячи і супроводжуються незначним фізичним напруженням, з енерговитратами до 120 ккал / годину), необхідно дотримуватися наступних 84 норм: коефіцієнт

відображення більше 0,4, тобто світлий фон; контраст об'єкта з фоном великий і середній при К більше 0,2 (згідно СНіП 11-4-79).

У полі зору користувача ПЕОМ має бути забезпечений відповідний розподіл яскравості. Відношення яскравості екрана до яскравості оточуючих його поверхонь не повинно перевищувати у робочій зоні 3:1 (СНіП 11-4-79).

У зв'язку з цим дисплей ПЕОМ повинен відповідати наступним вимогам:

- Яскравість свічення екрану не менше 100 кд/м;
- Мінімальний розмір світної точки для кольорового дисплея не більше 0,6 мм ;
- Контрастність зображення знаку – не менше 0,8; • Низькочастотне тремтіння зображення в діапазоні 0,05-1,0 Гц повинно знаходитися в межах 0,1 мм;
- Екран повинен мати покриття антивідблиску;
- Відеомонітор повинен бути обладнаний поворотним майданчиком, що дозволяє переміщати відеотермінал в горизонтальній і вертикальній площинах в межах 130-220 мм і змінювати кут нахилу на 10-15 мм.

Коефіцієнт відбиття світла матеріалами і обладнанням всередині приміщень має велике значення для освітлення: чим більше світла відбивається від поверхонь, тим вище освітленість. Коефіцієнт відображення відповідно повинен бути для: стелі 60-70%, стін 40-50%, підлоги 30%, для інших поверхонь 30-40%. Результати досліджень показують, що найбільшою мірою негативний фізіологічний вплив на операторів ПК пов'язаний з дискомфортними зоровими умовами через неправильно спроектоване освітлення. Згідно СНіП II-4-79 освітленість на горизонтальній площині робочого місця оператора ЕОМ повинна складати 400 лк при висоті цій площині 0,8 м над підлогою.

4.2.2 Вимоги до освітленості і повітряного середовища в робочій зоні

Світловий клімат визначає зоровий дискомфорт. Запобігти шкідливому впливу освітлення можна шляхом правильного підбору системи освітлення, джерел світла (за їх спектрального складу випромінювання), світильників. Коли штучне світло змішується з природним, рекомендується використовувати лампи за спектральним складом найбільш близькі до сонячного світла. Світильники слід вибирати з розсіювачами, а блискучі деталі освітлювального обладнання, що можуть потрапити в поле зору оператора, повинні бути замінені на матові. Розташовувати робоче місце, обладнане дисплеєм,

необхідно таким чином, щоб у полі зору оператора не потрапляли вікна або освітлювальні прилади; вони не повинні знаходиться і безпосередньо за спиною оператора. Вікна в приміщеннях з дисплеями обладнають шторами з коефіцієнтом відображення 0,5 ... 0,7, стіни фарбують матовою фарбою з коефіцієнтом відображення 0,4 ... 0,6. Світловий клімат може бути поліпшений шляхом встановлення спеціальних антивідблискових контрастних фільтрів, однак при виборі типу фільтра необхідно враховувати умови роботи з комп'ютером, оскільки оптимальні значення коефіцієнтів пропускання і дзеркального відображення фільтрів залежать від освітленості робочого місця і типу джерела світла. Враховуючи великий вплив освітлення на працездатність оператора при роботі з комп'ютером, проведемо розрахунок необхідної освітленості в приміщенні з дисплеями при наступних умовах: гігієнічна норма освітленості на горизонтальній поверхні на рівні робочого місця оператора – 400 лк; ширина приміщення – 7 м, довжина – 8 м, висота – 3 м. Коефіцієнт відбиття від стелі – 70, від стін – 50, від робочих поверхонь – 30. Повітряне середовище – нормальне (вміст пилу, диму й кіптяви не більше 5 мг/м³). Повітряне середовище в робочій зоні визначається мікрокліматом виробничого приміщення. Величини температури, відносної вологості та швидкості руху повітря на робочих місцях з дисплеями повинні відповідати 86 допустимим значенням, які встановлені ГОСТ 12.1.005-88 ССБТ для категорії робіт 1а (легкі фізичні роботи, вироблені сидячи і супроводжуються незначною фізичною напругою до 120 ккал/год.). Згідно з цим документом допустимі значення температури повітря в приміщенні становлять 19-25 °С, відносної вологості повітря – 55%, швидкості руху повітря на рівні особи – 0,1 м/С. При наявності досить комфортного робочого середовища атмосферний тиск по ГОСТ 21552-84 ССБТ може змінюватися від 84 до 107 кПа (630 ... 800 мм рт. ст.). Шум несприятливий для людини, особливо при тривалому впливі. В оператора це виражається в зниженні працездатності (наприклад, швидкість обробки тексту зменшується на 10-15%), у прискоренні розвитку зорового стомлення, зміну відчуття кольору, підвищенні витрати енергії (на 17%). Тривалий та інтенсивний шум значно знижує продуктивність праці і призводить до зростання кількості помилок у роботі. У відділі головного економіста шум може створюватися телефонними дзвінками та розмовами, системними блоками та клавіатурою ПЕОМ. Так само джерелами шуму можуть бути системи кондиціювання та вентилювання повітря, існують і зовнішні джерела шуму (наприклад, працюють агрегати на вулиці).

4.2.3 Допустимі рівні звуку на робочих місцях

Допустимі рівні звуку та еквівалентні рівні звуку на робочих місцях повинні відповідати вимогам «Санітарних норм допустимих рівнів шуму на робочих місцях» № 3223-85. Згідно з цими нормами в приміщенні, де працює користувач ПЕОМ для забезпечення оптимальної робочої середовища рівень шуму не повинен перевищувати 60 дБ. Основними заходами боротьби з шумом згідно з ГОСТ 12.1.029-80 ССБТ є ліквідація або ослаблення джерела шуму шляхом застосування звукопоглинаючих матеріалів у конструкціях механізмів, використання коштів звукопоглинання і раціональна планування виробничого приміщення.

Випромінювання ПК можуть бути небезпечними для здоров'я. Низькочастотні поля при тривалому опроміненні сидять біля ПК людей можуть привести до порушень самих різних фізіологічних процесів. Відповідно до ГОСТ 27016-86 і ГОСТ 27954-88 потужність дози рентгенівського випромінювання в будь-якій точці простору на відстані 5 см від екрану відеомонітора при 41 годинному робочому тижні не повинна перевищувати 100 мкР/год (0,03 мкР/с), а інтенсивність ультрафіолетового випромінювання – 10 Вт/м². В даний час випускаються вибухобезпечні відеомонітори. За способом захисту людини від ураження електричним струмом дисплеї виготовляються відповідно з 1-м класом захисту за ГОСТ 25861-84, тому кабель живлення дисплея має вилку з трьома виводами, один з яких заземлюючий.

Для забезпечення ПДУ чинників робочого середовища на робочих місцях у необхідних випадках використовуються спеціальні засоби захисту працюючих. Способи захисту бувають активними і пасивними. Способи активного захисту засновані на виявленні джерел несприятливих факторів і вплив на них. У випадках неможливості здійснення активного захисту застосовується пасивна, за якої джерела несприятливих факторів залишаються, але здійснюються заходи, спрямовані на попередження вплив цих факторів на людину. Пасивна захист може бути колективного та індивідуального. Розглянемо колективні засоби захисту оператора ПК.

Висока температура повітря негативно позначається на функціональному стані людини. Всі основні електронні блоки ПК мають вбудовані вентилятори для забезпечення стабільних температурних режимів їх функціонування, тому при створенні комфортних умов роботи особливу увагу необхідно приділити шляхам відводу повітря (припливно-витяжної вентиляції).

Для захисту від електростатичного потенціалу і, певною мірою, від електричної складової змінного електромагнітного поля (ЕМП) можуть бути

використані згадані вище антиблискові контрастуючі фільтри на екрани дисплеїв.

ВИСНОВОК

У процесі виконання випускної кваліфікаційної роботи був проведений огляд існуючих інтегрованих середовищ розробки, зроблений вибір однієї з них для використання в даній роботі. Розглянуто нова сучасна технологія XAMARIN, згідно заданої предметної області складено технічне завдання для розробки програмного продукту.

Розроблено переноситься бібліотека класів, яка містить основний функціонал додатка, клієнтська частина для платформ Android і Windows, а так же серверна частина програми у вигляді REST сервісу. Складено керівництво користувача для версій на обох платформах.

Розробка велася на мові C # в середовищі розробки Visual Studio 2015 Enterprise with XAMARIN на базі .NET Framework 4.5.2 і Android SDK API level 20-23.

У розділі Фінансовий менеджмент проаналізовано основні конкуренти, виявлені переваги і недоліки продукту в порівнянні з конкурентами, і на основі цього обрано шлях подальшого розвитку програми.

У розділі охорона праці розглянуті питання організації робочого місця розробника, вимоги до приміщення, мікроклімату в робочій зоні, ергономіці робочого місця, забезпечення електро і пожежної безпеки.

По завершенню роботи можна сказати, що поставлена мета була досягнута, завдання вирішені.

Список використаних джерел

1. Введення в інтегроване середовище розробки Eclipse. [Електронний ресурс] URI: http://www.javaportal.ru/java/ide/intro_eclipse.html, вільний. - Загл. з екрану. - Яз. рус. Дата звернення: 13.02.2016
2. About the Eclipse Foundation. [Електронний ресурс] URI: https://www.eclipse.org/org/, вільний. - Загл. з екрану. - Яз. англ. Дата звернення: 13.02.2016
3. NetBeans IDE Features. [Електронний ресурс] URI: https://netbeans.org/features/index.html, вільний. - Загл. з екрану. - Яз. англ. Дата звернення: 13.02.2016
4. NetBeans IDE - універсальна інтегрована середовище розробки додатків. [Електронний ресурс] URI: http://hightech.in.ua/content/artnetbeans-ide. - Загл. з екрану. - Яз. рус. Дата звернення: 13.02.2016
5. Visual Studio Enterpise. [Електронний ресурс] URI: https://www.visualstudio.com/products/visual-studio-enterprise-vs. - Загл. з екрану. - Яз. рус. Дата звернення: 13.02.2016
6. Детально про XAMARIN. [Електронний ресурс] URI: https://habrahabr.ru/post/188130, вільний. - Загл. з екрану. - Яз. рус. Дата звернення: 13.02.2016
7. Центр розробників XAMARIN. [Електронний ресурс] URI: http://developer.xamarin.com, вільний. - Загл. з екрану. - Яз. англ. Дата звернення: 13.02.2016
8. Центр розробників ANDROID. [Електронний ресурс] URI: http://developer.android.com, вільний. - Загл. з екрану. - Яз. рус., англ. Дата звернення: 13.02.2016
9. Центр розробників MSDN. [Електронний ресурс] URI: http://msdn.microsoft.com, вільний. - Загл. з екрану. - Яз. рус., англ. Дата звернення: 13.02.2016
10. Шілдт Герберт. Повний довідник по C # [Текст]: довідник / Шілдт Герберт. - Вільямс, 2005. - 752с.
11. Bewis T. C # Design Pattern Essentials. - NY: Ability First Limited, 2012. - 264 p.
12. Freeman A. Pro ASP.NET MVC 5 (Expert's Voice in ASP.Net). - NY: Apress, 2013. - 832 p.11 3. Пратт Т., Зелковіц М. Мови програмування: розробка і реалізація / під заг. ред. О. Матросова. - СПб .: Пітер, 2002. -688 с.
13. Gomaa H. Software Modeling and Design: UML, Use Cases, Patterns,and software Architectures. - NY: Cambridge University Press, 2011. - 578 p.

14. НАПБ А.01.001-2004. Правила пожежної безпеки в Україні. − К.: Міністерство України з питань надзвичайних ситуацій, 2004.
15. Навакатікян О.О., Кальниш В.В., Стрюков С.М. Охорона праці користувачів комп'ютерних відео дисплейних терміналів. − К.: 1997. − −400 с.
16. ДСанПіН 3.3.2.007-98. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин. – К.: МОЗ України,1998.
17. Посібник по БЖД. [Електронний ресурс] URI: http://www.studfiles.ru/preview/434015, вільний. - Загл. з екрану. - Яз. рус.
18. Hermes D. Xamarin Mobile Application Development (2015)

Додаток А.

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ ІВАНА ПУЛЮЯ

КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку проекту

«Мобільна гра «Знайди пару»»

Розробники:

виконавець ст. гр. СПмз-61

Луків Сергій Ярославович

_____

(підпис)

керівник проекту

Петрик Михайло Романович

_____

(підпис)

Тернопіль 2020

ЗМІСТ

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану підготовки бакалаврів за спеціальністю 121 «Інженерія програмного забезпечення».

Тема проекту: « Мобільний додаток «Знайди пару» ».

Термін виконання: до 23.12.2020р.

2 ПРИЗНАЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

Інформаційна система призначена для автоматизації тренувань користувача.

Інформаційна система буде корисною в сферах спорту, здоров’я, рекреаційної медицини та стане невід’ємною частиною у саморозвитку.

Інформаційна система дозволить здійснювати контроль тренувань, навантаження на різні групи м’язів, розклад, отримувати звітність про тренування та навантаження.

3 ВИМОГИ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Функціональні вимоги

Система повинна передбачати одну роль:

- користувач

Для користувачів система надає доступ до повного переліку функцій.

- відображення більшості вправ, які можна виконувати у залі;
- опис виконання вищезазначених вправ та їх графічне зображення;
- можливість створення власних програм тренувань;
- графік тренувань;
- можливість використання вже готових програм;
- вибір системи тренувань для будь-якої групи м’язів;
- діаграма відвідувань.

Набір даних функцій дозволяє користувачу створити оптимальну для його структури тіла систему тренувань, дізнатись про те, як вірно виконувати вправи із вибраної системи.

3.2 Технічні вимоги

Вимоги до клієнтської частини: ОС Android, не старіше 4.2 версії, інтуїтивно зрозумілий, не перевантажений інтерфейс: будь-яка бажана дія зі сторони користувача повинна досягатися в межах трьох дотиків до екрану.

3.3 Програмні вимоги

Використання СУБД: SQLite

Розробка серверної частини: Java

Розробка клієнтської частини: XML

Додаткові вимоги: використання алгоритму криптозахисту при передачі даних по відкритих каналах зв'язку, ...

4. ЕТАПИ РОЗРОБКИ

Розробка інформаційної системи проводиться в наступному порядку:

- аналіз предметної області, виявлення акторів та варіантів використання системи;
- вибір засобів розробки та архітектури системи;
- проектування бази даних системи;
- розробка програмного забезпечення системи;
- тестування інформаційної системи на реальних даних;
- оформлення супровідної документації;
- здача проекту.

Результати виконання кожного етапу проекту погоджуються з керівником проекту.

5. СУПРОВІДНА ДОКУМЕНТАЦІЯ

Для інформаційної системи повинні бути розроблені наступні документи:

- пояснювальна записка до проекту;
- презентація проекту;
- рецензія на проект;
- диск з проектом.

Пояснювальна записка до проекту оформляється згідно діючих вимог до нормоконтролю проектів.

6. ПОРЯДОК ЗДАЧІ ПРОЕКТУ

Розроблена інформаційна системи повинна відповідати вимогами, що складаються з перерахованих у п.3.1 цього документу характеристик.

Для здачі проекту необхідно підготувати весь перелік документів зазначений у п.5 цього документу.

Приймання проекту проводиться спеціально створеною комісією в термін зазначені в п.1 цього документу.

7. ВІДМІТКИ ПРО ВИКОНАННЯ ЕТАПІВ ТА ЗМІНИ В ПРОЕКТІ

Назва етапу	Відмітка*
Аналіз предметної області	
Архітектура системи	
Проектування база даних	
Використання системи	
Супровідна документація	

* відмітки про виконання етапу ставляться керівником проекту

Луків С.Я., магістр другого року навчання кафедри програмної інженерії Тернопільський національний університет імені Івана Пулюя.

Розробка мобільної гри «знайди пару»

Lukiv SY, master of the second year of study of the Department of Software.

Development of a mobile game "find a couple"

Популярність мобільних засобів появилася через велику взаємодію з людино так як мобільні пристрої за допомогою сучасних технологій здатні в великому обсязі задовольнити інформаційні потреби людей. Через свою зручність та гнучкість вони замінили багато пристроїв, що були необхідними для користувача та колись здавалися незамінними, такі як стаціонарні телефони, відеокамера, навігатор, радіо приймач, фотокамера, блокнот, календар, відеокамера. Також дані пристрої зайняли і велику нішу і сфері відео ігор. Ні для кого не секрет, що відео ігри міцно зайняли свою позицію в сучасній індустрії розваг. Існують спроби виділити комп'ютерні ігри як окрему область мистецтва, поряд з театром, кіно і т.п. Розробка ігор може виявитися не тільки захоплюючим, але й прибутковою справою. А в зв'язку з ростом кількості мобільних пристроїв і уніфікації способів розробки додатків для кожного з них, з'являється широка перспектива комерційної розробки. Ринок мобільних пристроїв, а відповідно і їх користувачів стрімко зростає і потрібно все більше і більше контенту для них: програми, ігри, утиліти, спрямовані на повне розкриття потенціалу мобільних пристроїв. Цим і визначається актуальність обраної теми. Метою роботи є розробка кроссплатформенного клієнтсерверного ігрового програми «Знайди пару». Об'єктом дослідження є технологія кроссплатформенной розробки на базі XAMARIN, що дозволяє розробляти Кросплатформені додатки в середовищі MS Visual Studio на мові C# і компілювати їх код для кожної платформи. Предметом дослідження є проектування переносний кроссплатформенной бібліотеки класів, розробка серверної частини, інтерфейсу на платформах Windows і Android. Практична новизна: досліджувана технологія абсолютно нова і стрімко розвивається, з

кожним місяцем яка приваблює все більшу кількість розробників. Тим самим наближаючи і об'єднуючи безліч існуючих платформ воєдино для зручної, швидкої та якісної розробки додатків. Практична значимість результату виконаної роботи: розроблений додаток можна використовувати в комерційних цілях, як для продажі, так і заробітку на рекламі (в перспективі зростання кількості користувачів).

Література

1. Hermes D. Xamarin Mobile Application Development (2015)

Додаток Б. Диск з програмою