Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЕКТ

на тему:

«Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»

Студента 2 курсу, 6м групи,
спеціальності 121
«Інженерія програмного
забезпечення»,
спеціалізації «Інженерія
програмного забезпечення» Бабича Володимира
Олеговича

_____
підпис студента

Науковий керівник
кандидат технічних наук
доцент кафедри інженерії
програмного забезпечення та
кібербезпеки Цензура Микола
Олександрович

_____
підпис керівника

Гарант освітньої програми
Завідувач кафедри інженерії
програмного забезпечення та
кібербезпеки
доктор технічних наук, професор Криворучко Олена
Володимирівна

_____
підпис керівника

КИЇВ – 2020

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
8 листопада 2019 р.

Завдання
на випускний кваліфікаційний проект студентові

Бабича Володимира Олеговича
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проекту «Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»
Затверджена наказом ректора від 24.12.2019 №4440
2. Строк здачі студентом закінченої проекту 01 грудня 2020
3. Цільова установка та вихідні дані до проекту
Мета проекту створення мобільного додатку за допомогою якого можна поліпшити виховання дітей.
Об'єкт дослідження процес функціонування дошкільних закладів.
Предмет дослідження методи та алгоритми, як засоби підготовки до безперервно-контрольованого процесу виховання.

4. Консультанти проекту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проекту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ОГЛЯД ТА ЗАСОБИ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

1.1. Ключові моменти при розробці мобільних додатків

1.2. Перспективи розвитку мобільного середовища

1.3. Найпопулярніші мобільні операційні системи та їх характеристика

РОЗДІЛ 2. ПОТРЕБА ЗАКЛАДУ ДОШКІЛЬНОЇ ОСВІТИ У СТВОРЕННІ МОБІЛЬНОГО ДОДАТКУ

2.1. Стан закладів дошкільної освіти в Україні

2.2. Вирішення проблем батьків та приватних садків

2.3. Огляд потрібного обладнання

РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ IOS.

3.1. Засоби розробки

3.2. Огляд мов програмування для розробки

3.3. Розробка архітектури розробляемого проекту

3.4. Розробка серверної частини

3.5. Розробка клієнтської частини

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

6. Календарний план виконання проекту

№ по р.	Назва етапів випускного кваліфікаційного проекту	Строк виконання етапів проекту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проект*	*20.09.2019*	*20.09.2019*
2.	*Розробка та затвердження завдання на проект магістра*	*08.11.2019*	*08.11.2019*
3.	*Вступ та перелік літературних джерел*	*24.01.2020*	*24.01.2020*
4.	*Наукова стаття*	*01.09.2020*	*01.09.2020*
5.	*Технічне завдання*	*28.02.2020*	*28.02.2020*
6.	*Розділ 1. Огляд та засоби розробки мобільного додатку*	*25.06.2020*	*25.06.2020*
7.	*Розділ 2 Потреба закладу дошкільної освіти у створенні мобільного додатку*	*07.09.2020*	*07.09.2020*
8.	*Розділ 3. Розробка мобільного додатку для операційної системи IOS*	*19.10.2020*	*19.10.2020*
9.	*Програма та методика тестування*	*21.10.2020*	*21.10.2020*
10.	*Керівництво користувача*	*23.10.2020*	*23.10.2020*
11.	*Висновки та пропозиції*	*30.10.2020*	*30.10.2020*
12.	*Здача випускного кваліфікаційного проекту на кафедру (перша перевірка)*	*05.11.2020*	*05.11.2020*
13.	*Підготовка автореферату та презентації доповіді*	*05.11.2020*	*05.11.2020*
14.	*Попередній захист випускного кваліфікаційного проекту*	*25.11.2020-27.11.2020*	*25.11.2020-27.11.2020*
15.	*Зовнішнє рецензування випускного кваліфікаційного проекту*	*01.12.2020*	*01.12.2020*
16.	*Підготовка до публічного захисту випускного кваліфікаційного проекту*	*10.12.2020-11.12.2020*	

7. Дата видачі завдання «8» листопада 2019 р.

8. Науковий керівник випускного кваліфікаційного проект
Цензура М.О.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Криворучко О.В.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Бабич В. О.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проекту

Науковий керівник випускного кваліфікаційного проекту
Цензура М.О
(підпис, дата)

Відмітка про попередній захист
(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проект

Випускний кваліфікаційний проект студента Бабич В.О.
(прізвище, ініціали)
може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми Криворучко О. В.
(прізвище, ініціали, підпис)

Завідувач кафедри Криворучко О. В.
(підпис, прізвище, ініціали)

«_____» _____ 20___р

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці мобільного додатка для закладу докшкільної освіти, що дозволить поліпшити виховання дітей, та забезпечити їм безпечне середовище.

В результаті порівняльного аналізу аналогічних рішень визначено вагомі недоліки подібних iOS-додатків, що були прийняті до уваги та усунені у нашій розробці.

Розробка виконана у середовищі розробки Xcode. Обрана мова програмування – Swift. Окремі елементи мобільного додатка розроблені із використанням Interface Builder, відповідно до стандартів розробки iOS - додатків.

Готовий програмний комплекс ChildSaver було успішно протестовано відповідно до функціональних вимог.

Ключові слова: iOS, мобільний додаток, Xcode, Swift.

ABSTRACT

In accordance with the purpose of the study, the work is devoted to the development of a mobile application for preschool education, which will improve the upbringing of children and provide them with a safe environment.

As a result of comparative analysis of similar solutions, significant shortcomings of such iOS-applications were identified, which were taken into account and eliminated in our development.

Development is performed in the Xcode development environment. The selected programming language is Swift. Individual elements of the mobile application are developed using Interface Builder, in accordance with the standards of iOS application development.

The finished ChildSaver software package was successfully tested in accordance with the functional requirements.

Key words: iOS, mobile application, Xcode, Swift.

ЗМІСТ

					КНТЕУ 121 06з-1.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		19.10.20	«Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»	Стадія	Аркуш	Аркушів
Керівник		Цензура М.А.		19.10.20		З	2	78
Гарант		Криворучко О.В.		19.10.20		*Факультет інформаційних технологій 2м курс, 6з група*		
Розробив		Бабич В.О.		19.10.20	*Зміст*			

ВСТУП

Актуальність: Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій за останні роки. Перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці та в будь-якій ситуації.

Об'єктом дослідження є процес функціонування дошкільних закладів.

Предметом дослідження є методи та алгоритми, як засоби підготовки до безперервно-контрольованого процесу виховання.

Мета роботи полягає у створенні мобільного додатку за допомогою якого можна поліпшити виховання дітей.

Результатом роботи є створення мобільного додатку за допомогою якого батьки матимуть змогу контролювати своїх дітей, та втручатись у процесі виховання.

Сучасний рівень розвитку інформаційних технологій відкриває перспективи використання користувачами принципово нових сервісів в вихованні — «мобільна присутність у процесі виховання», що надає можливість контролювати незалежно від місця і часу, виховний процес та поведінку дитини у дошкільних закладах.

Мобільне присутність у процесі виховання є новою виховною парадигмою, на основі якої створюється інше виховне середовище, яке робить сам процес виховання безперервно-контрольованим та мотивує працівників дошкільних закладів відповідально ставитись до своїх обов'язків.

					КНТЕУ 121 06з-1.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		24.01.20	«Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»	Стадія	Аркуш	Аркушів
Керівник		Цензура М.А.		24.01.20		В	3	78
Гарант		Криворучко О.В.		24.01.20		*Факультет інформаційних технологій 2м курс, 6з група*		
Розробив		Бабич В.О.		24.01.20	*Вступ*			

Актуальність даної теми зумовлюється тим, що мобільні технології входять в життя людей досить стрімко. Кожний батько має мобільний телефон, тому одним із завдань є впровадження відповідних мобільних додатків, які допоможуть безперервно-контрольовати процес виховання та ставлення працівників до дитини.

Результатом роботи є розроблений та впроваджений мобільний додаток для платформи iOS.

Мета обумовила вибір наступних завдань:

- дослідити сутність мобільного навчання та шляхи його впровадження в систему освіти;
- провести аналіз методів та технологій розробки мобільних додатків;
- обрати технологію розробки для створення мобільного додатку;
- сформувати функціональну структуру мобільного додатку для підготовки до зовнішнього незалежного оцінювання;
- розробити програмний продукт та провести його тестування.

По даним Gartner, у 2018 році по всьому світу було продано більше 1.5 млрд смартфонів. Станом на 2018 р. більше ніж одного мілионну додатків було розроблено для iOS. Аналіз показав, що більше 70% мобільних розробників використовували платформу iOS для розробки та публікації додатків.

Об’єктом розробки магістерської роботи є мобільний додаток, створений на ОС iOS, який зберігає інформацію про особливості функціонування дошкільного закладу, а розробка додатку є під ОС iOS є найбільш актуальним процесом. ОС iOS, це повноцінна операційна система, в основі якої покладене ядро POSIX. Самі перші версії знайшли соє застосування у сегменті стільникових телефонів, включаючи смартфони. Однака повний спектр і великі функціональні можливості iOS дозволяють створювати додатки, які виходять далеко за сегмент стільникових телефонів.

У сучасних умовах розробка додатків у більшості випадків виде до потреби використання інтегрованих середовищ розробки (ICP). Вони мають безперечні переваги, саме: процес компіляції, збірка і запис додатку зазвичай автоматизовані. Сучасний ICP «iOS розробника» повинні підтримувати мови програмування: Swift, Objective C, C++, а також інші мови віртуальної машини, iOS, які регулярно використовуються. ICP повинні бути сумісні з будь якими збірками систем контроля версій, наприклад, Ant, Maven або Gradle.

Одним з найбільш популярних ICP для розробки під ОС iOS є Xcode. З точки зору можливостей і ціни Xcode безкоштовный для користувачів MacOS.

В сучасному світі набуває все більше обертів розвиток новітніх технологій та їх використання серед населення. Особливо це стосується молодих батьків, які мають велику потребу у побудові своєї кар'єри та активний відпочинок. Це залишає мало часу на виховання дітей, що змушує відправляти їх у дошкільні заклади.

Технічним завданням на проектування мобільного додатку контроля над діяльністю дошкільного закладу із сторони батьків. Цей додаток повинен допомогти батькам контролювати загальний стан у заклад: територію навколо закладу, відношення між дітьми та вихователям. Також важливо зробити комунікації між батьками та вихователями, на випадок екстреної ситуації або залишати коментарі.

У якості засобів розробки були вибрані: мова програмування Swift, сумісний вебхостинг, який буде створений на базі Apache2, PHP5 та My SQL. Специфічні вимоги до розробити додатку: поліпшити комунікації батьків з адміністрацією закладу і вихователями; контролювати всі події, що відбуваються у приміщеннях; оперативно встановлювати аудіо-відео зв'язок з працівниками закладу.

РОЗДІЛ 1.
ОГЛЯД ТА ЗАСОБИ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

1.1. Ключові моменти при розробці мобільних додатків

Найбільш суттєві відмінності між мобільними програмами та настільними програмами, ймовірно, пов'язані з тим, як люди ними користуються. Найбільша категорія користувачів використовує комп'ютер головним чином для таких видів роботи: перегляд ресурсів в Інтернеті, робота з документами (обробка текстових документів, електронних таблиць, малюнків тощо) та спілкування (електронна пошта, обмін миттєвими повідомленнями тощо). У типових випадках усі ці дії є довготривалими, дослідницькими за своєю природою, і користувач періодично переходить від однієї з цих основних видів діяльності до іншої.

У більшості випадків методи використання настільних та мобільних версій одного додатка різняться, хоча вони доповнюють один одного. Як правило, робота користувача з мобільним додатком - це характер короткочасних сеансів взаємодії, які або перериваються ззовні, або переривають сеанс. Як правило, користувач або відповідає на спробу перервати свою роботу, або використовує пристрій для негайної передачі запиту конкретній особі чи процесу. Типова практика використання сучасних мобільних телефонів може бути чудовою ілюстрацією до цього. Коли ви здійснюєте телефонний дзвінок або надсилаєте текстовий SMS, ваш абонент отримує сигнал переривання; якщо вони зателефонують або відправлять вам SMS, ваша робота буде перервана.

Ця ж модель використання передається будь-якій іншій програмі, що працює на мобільних пристроях. Мобільний додаток можна вважати успішно розробленим лише в тому випадку, якщо ви використовуєте його так само просто і природно, як спілкування з кимось по телефону або обмін SMS-повідомленнями.

Крім того, мобільні додатки мають сильну спеціалізацію, яка забезпечує найбільш зручний доступ до обмеженого набору конкретних функцій, що відрізняють пристрої від універсального дослідницького середовища, створення якого характерне для успішних настільних додатків. Оскільки мобільні пристрої часто керують однією рукою або невеликим екраном, важливо, щоб користувач міг швидко ідентифікувати та швидко знайти потрібну йому інформацію чи інструменти. Можливість швидкого навігації по декількох наборах ключових інструментів є важливим показником якості розробки мобільних додатків.

Робота на робочому столі зазвичай проводиться під час тривалих робочих сесій. Найпоширенішими є сеанси спостереження, але вони часто затримуються на кілька годин. У процесі тривалої роботи ідеї формуються та розвиваються, а сама робота є ітеративною з її цілісною прибутковістю до вже пройдених етапів. Тому не важливіше скоротити тривалість підготовки до роботи, а забезпечити користувача багатим набором різних інструментів, які йому можуть знадобитися в процесі дослідження та обробки відповідного інформатора.

Лептопи в основному використовуються так само, як і настільні комп'ютери, але в цьому випадку ви вже можете побачити ознаки, характерні для використання мобільних пристроїв. Більше уваги приділяється скороченню часу запуску (або пробудження) програми та можливості користувача швидко перейти до роботи, яку він виконував до кінця попереднього робочого сеансу.

У той же час характер використання лептопів ближче до настільних комп'ютерів, ніж до мобільних пристроїв, що частково пов'язано з близькістю фізичних розмірів відповідних категорій обладнання.

Що стосується мобільних пристроїв, то такі дії користувач може виконувати набагато швидше, а в типових випадках це займає від декількох секунд до декількох хвилин. Такі операції, як телефонні дзвінки, графіки зустрічей або повідомлення про надсилання та читання, виконуються під час часто повторюваних, але коротких сесій. Для мобільних пристроїв навіть ігри - це короткочасні дії. Як правило, мобільні пристрої використовуються для ігор протягом коротких періодів часу між іншими операціями. Ігри розроблені, щоб допомогти людям скоротити час очікування поїздів, залишитися в аеропорту або чекати партнера, який спізнюється на зустріч з вами. Це короткотермінові фактори використання та можливість негайно отримати доступ до необхідних коштів та змусити нас підтримувати мобільні пристрої в стані «постійного" або в стані "негайної готовності до включення", протягом якого потрібен час для доступу пристрій, який слід порівнювати з часом.

Важливо зазначити, що, незважаючи на короткі робочі сесії, дані, з якими працює користувач, часто є довгостроковими. Користувач мобільного пристрою може захотіти продовжити гру з місця її переривання протягом попереднього сеансу.

Під час переговорів по телефону або під час особистих зустрічей вам часто доводиться переходити до розкладу зустрічей і вказувати дату та час наступної зустрічі. Відповідні дані залишаються актуальними протягом тривалого часу і повинні зберігатися тривалий час, але сама програма використовується протягом коротких періодів часу з прямим доступом до інформатора. Порівняльні та контрастні мобільні пристрої та їх програми зі своїми настільними та серверними аналогами, у цьому розділі ми зайняли багато часу.

1. Час початку. Важливою особливістю мобільних додатків є час їх запуску. Оскільки мобільні пристрої зазвичай використовуються часто і протягом коротких періодів часу, можливість швидкого запуску мобільного додатка є обов'язковою. Рекомендується дотримуватися правила mnemonic, згідно з яким тривалість сеансу користувача з додатком повинна значно перевищувати тривалість запуску цієї програми. Для настільних додатків робочі сесії тривають довше, тому користувачі готові миритися з більш тривалими періодами очікування. Для мобільних додатків тривалість робочих сесій коротша, і тому необхідно, щоб тривалість періоду запуску програми була відповідно коротшою.

2. Швидкий відгук. Беручи мобільні пристрої як слухняні механічні іграшки, люди очікують, що вони поводяться правильно. Коли користувач діє на маніпулятор, натискає кнопку або іншим чином маніпулює керуванням, він покладається на фізичну реакцію зі свого боку. Не отримуючи негайної реакції від пристрою, нетерплячий користувач нервує і негайно повторює спробу виконати необхідні операції.

Якщо ви спробуєте знову маніпулювати контролем, натискання або інші подібні дії будуть оброблені вашою програмою або, помилково, іншою програмою, це може спричинити проблеми.

Варіант для такого підтвердження - виконати запитувану операцію. По-друге, у зв'язку з цим є підтвердження того, що запит був отриманий та оброблений у фоновому режимі, тоді як заявка зберігає можливість приймати інші запити. Третє місце належить дисплею резервного курсору, який буде представлений користувачеві під час обробки запитів; програма не відповідає на подальші запити, але користувач знає, що робота з обслуговування її запиту була розпочата і виконується.

Найгірше з варіантів - коли користувач не бачить відповідної реакції з пристрою, і він може лише здогадуватися, приймається його запит чи ні.

3. Зосередження на конкретних завданнях. Ще одна характеристика мобільного додатку, яка визначає його успіх, - це ступінь його спеціалізації. Для програми повинен бути чітко визначений набір завдань, ефективне рішення якого він повинен забезпечити; операції повинні виконуватися швидко і вимагати від користувача мінімальної кількості кліків, кліків чи інших дій.

З цієї причини пристрої часто надають спеціальні кнопки, які відповідають певним завданням, які цей пристрій може вирішити (наприклад, кнопка для переходу безпосередньо до бази даних, в якій зберігається контактна інформація, кнопка для перегляду календаря зустрічей або кнопка для читання штрих-код і перенести його на певну програму.

Ви повинні докласти зусиль, щоб більш точно визначити коло загальних завдань, які повинні вимагати мінімальних дій користувача. Ця рекомендація повинна керуватися як функціями програми високого рівня, так і завданнями низького рівня, які користувачі зазвичай повинні вирішувати в процесі роботи з цією програмою. Наприклад, якщо мобільний додаток забезпечує швидке введення дати, вам потрібно подбати про максимальне спрощення (і прискорення) зазначеного процесу та вимірювання його ефективності. Якщо ваш мобільний додаток повинен мати можливість вказати конкретне місце за допомогою карти міських вулиць, щоб користувач міг вибрати маршрут до кінцевого пункту призначення, вам потрібно переконатися, що ви можете зробити це якнайшвидше.

Однією з поширених помилок, які роблять розробники під час створення програм для мобільних пристроїв, є те, що програмний код намагається зробити його якомога коротшим, щоб мінімізувати розмір програми.

Хоча такі цілі благородні, їх не слід досягати за рахунок зниження продуктивності користувачів.

4. Взаємодя із зовнішніми джерелами інформації. Дуже важливо розуміти, що створення чудового мобільного додатку - це не просто запис коду, який правильно виконується на пристрої. Ви також не повинні забувати про зовнішнє програмне забезпечення, з яким взаємодіє ваш мобільний додаток. Програма повинна належним чином враховувати характеристики джерел інформації, що обслуговує мобільні пристрої, та надавати інформацію у формі, що відповідає цьому пристрою. Наприклад, ви можете надати послуги електронної пошти для мобільних пристроїв. Багатофункціональні поштові програми вимагають наявності відповідного програмного забезпечення як на стороні сервера, так і на стороні клієнта. Клієнт отримує доступ до сервера для отримання інформації про отримані повідомлення та завантаження відповідного вмісту за місцевими винятками. Оскільки мережеві з'єднання, що використовуються мобільними пристроями, мають неправильний доступ, обмежену пропускну здатність і, у багатьох випадках, більш високу вартість, ніж ті, що використовуються на персональних комп'ютерах, сервіси серверної пошти повинні бути адаптовані до цих вимог. Зазвичай це досягається шляхом надання серверу інструментів для обмеження кількості отриманого вмісту або фільтрів, які передають лише ту інформацію, яка може знадобитися користувачеві мобільного пристрою. Для підтримки ефективних сценаріїв при обслуговуванні мобільних пристроїв може знадобитися розширити функціональність тих серверних служб, які спочатку були призначені для обслуговування персональних комп'ютерів. Крім того, для його впровадження слід передбачити наступні механізми зміни налаштувань конфігурації програми на серверах, настільних комп'ютерах та мобільних пристроях, за допомогою яких користувачі можуть ідентифікувати свої інформаційні запити та налаштовувати інформаційні фільтри відповідно.

До відповідних вимог. Дизайн мобільних додатків далеко не обмежений фізичними межами самих мобільних пристроїв.

5. Той самий стиль інтерфейсу. Враховуючи, що будь-який мобільний пристрій відрізняється

компактність та самодостатність, бажання користувачів працювати з інтерфейсом, який не залежить від характеру завдань, вирішених пристроєм, здається цілком природним. Кожен мобільний пристрій має свій функціональний колір. Типовий добре розроблений додаток сприймається не як окремий набір інструментів, а як гармонійне розширення самого пристрою.

З цієї причини при розробці додатків для мобільних пристроїв дуже важливо дотримуватися того ж стилю. Шляхи запуску та зупинки програм, переміщення інтерфейсу та впорядкування діалогів між користувачами слід ретельно розглядати окремо для кожного типу пристроїв. Користувачі мобільних пристроїв підсвідомо адаптуються до метафор користувальницького інтерфейсу, а будь-які відхилення від звичайних шаблонів викликають у них відчутні незручності. Що стосується мобільних пристроїв, часто існує лише один спосіб вирішити цю проблему, і користувач мимоволі звикає до певного методу. Набагато краще мати чотири різні версії програми, кожна з яких відповідає певній метафорі користувача для інтерфейсу певного пристрою, ніж одна загальна програма, яка не може бути належним чином інтегрована в жоден цільовий пристрій. Відмінності в архітектурі комп’ютера.

Настільні комп'ютери OZU та довготривала пам'ять розділені. У той же час пристрої оперативної пам’яті часто використовуються як робоча пам'ять програми і як проміжна або довготривала пам'ять. пам'ять все частіше використовується як довготривалі сховища даних, як правило, у вигляді знімних карт пам'яті.

1.2. Перспективи розвитку мобільного середовища

Формування самого мобільного середовища тісно пов'язане з розвитком мобільних технологій, що дозволило створити мобільні пристрої, як ми їх бачимо зараз. Активно на формування цього середовища впливало поява загальнодоступного мобільного Інтернету та його швидке впровадження, проникнення в маси. Саме він дозволив перетворити саме мобільне середовище з банального способу спілкування в нового учасника створення, формування та розповсюдження інформації. І сьогодні ми спостерігаємо активний розвиток мобільної журналістики, активізацію ролі особистості у спілкуванні, здатність стати активним учасником її та впливати на інших.

У той же час мобільне середовище також призвело до швидкого збільшення кількості інформації та рівня її споживання. За прогнозами IDC, починаючи з 2011 року, кількість даних про планету подвоюється кожні два роки. Такий різкий стрибок значною мірою пов'язаний зі збільшенням доступності мобільного Інтернету у багатьох країнах. Зараз звичайний користувач може виробляти та споживати данні майже цілий день та скрізь. Якщо Інтернет дозволив отримувати інформацію, насправді, в декількох хвилинах ходьби, то з появою мобільних технологій вся інформація знаходиться в кишені користувача.

Сам термін мобільне середовище включає всі портативні пристрої, однак фахівці з розробки найчастіше звужують його виключно до наступного списку мобільних пристроїв: телефони, смартфони, фаблі, КПК, планшети, портативні пристрої (розумні годинники, браслети) та (повторно) електронні книги. Це звуження пояснюється популярністю та пріоритетом розвитку цих пристроїв.

Серед особливостей мобільного середовища можна виділити:

- ефективність (найвищий ступінь у відкритому доступі);
- нижня одиниця трафіку на одного користувача в порівнянні з настільними пристроями;

- найвища активність користувачів як у споживанні, так і в створенні контенту;
- низька конкуренція на ринку;
- аудиторія користувачів найбільш вигідна для реклами, оскільки він більш платоспроможний і активний у порівнянні з аудиторією настільних пристроїв.

Ви можете визначити додаток через термін комп'ютерної програми: послідовність інструкцій, призначених для виконання пристроєм управління комп'ютером. При цьому необхідно уточнити, що ця послідовність реалізована на мобільному пристрої (телефон, смартфон, планшет тощо).

Процес створення мобільного додатку передбачає його розвиток, тобто процес, при якому розробляються програми для невеликих портативних пристроїв, таких як КПК, смартфони або мобільні телефони. Ці програми можуть бути попередньо встановлені на пристрої під час виробництва, завантажені користувачем за допомогою різних платформ для розповсюдження програмного забезпечення або бути веб-додатками, які обробляються на стороні клієнта або сервера.

Розробники програм можуть пропонувати та публікувати свої програми в магазинах додатків із можливістю заробляти від розподілу доходів від продажу. Найвідоміші - AppStore, де лише затверджені програми можна розповсюджувати та запускати на пристроях iOS, та Google Play, програми, в яких працюють на пристроях з ОС Android [1.]

Перші спроби забезпечити доступ до новин через мобільний телефон були зроблені ще до періоду активного розповсюдження мобільного Інтернету. Так, на початку 2000-х численні видавці в розвинених країнах, таких як BBC і El País [2], багато німецьких [3] та шведських видань [4], використовували примусові сповіщення читачів через SMS. Кілька досліджень ілюструють динамічні зміни в мобільних новинах, що відбулися в країнах, що розвиваються, повідомляючи про поширення служб оповіщення новин у Бразилії [2], Китаї та кількох африканських країнах.

Дві китайські газети із засобів масової інформації Yunnan Daily Press Group запустили службу оповіщення про новини через SMS в 2002 році. Завдяки цій службі під час піку в січні 2005 року кількість абонентів на цю послугу становила 290 тисяч абонентів.

Однак тоді підписка почала різко скорочуватися, і лише за рік кількість абонентів впала до 120 тисяч.

Відповідь на таку кризу мобільних послуг було покращення інших. Розвиток мобільного Інтернету зробив його більш доступним для широкого кола користувачів, а виробники пристроїв надали можливість взаємодіяти з телефоном безпосередньо за допомогою сенсорного екрану, що спричинило певний бум у розвитку мобільної технології.

Паралельно зі швидким розповсюдженням сенсорних мобільних пристроїв виробники контенту, насамперед засоби масової інформації, відчули величезне зростання впливу мобільної платформи в цілому. Так, перше покоління iPhone було рекламовано разом із The New York Times і активно просувалося самою публікацією. Спочатку засоби масової інформації надавали доступ до їх вмісту з мобільних браузерів, таких як браузер Safari для iPhone, і лише потім почали розробляти так звані рідні програми новин. Завдяки ранньому випуску iOS на ринок багато світових ЗМІ вперше потрапили на мобільну платформу Apple, і лише після комерційного запуску Android у вересні 2008 року багато видавців вийшли на ринок від Google, багато в чому завдяки зусиллям різних виробники телефонів, наприклад, HTC, LG та. Платформи Blackberry та Symbian втратили всілякі ринкові позиції і давно не сприймаються як потенційні платформи розвитку. Третьою силою протистояння Apple і Google стала Microsoft з мобільною операційною системою Windows Phone (спочатку Windows Mobile) [6, с. 9-10.]. Міжнародна асоціація маркетингу новин (INMA) опублікувала тематичне дослідження "Схвалення стратегій мобільних новин" у червні 2012 року [7.]

Основна ідея звіту полягає в тому, щоб підкреслити як схожість, так і відмінності щодо сприйняття та дій різних представників видавництва новин у Європі та Північній Америці. Більшість респондентів кажуть, що споживання новин з мобільних пристроїв дає цінну можливість отримати доступ до своєї аудиторії в будь-який час і в будь-якому місці.

Це змусило певну частину новинних публікацій інвестувати в мобільні технології та послуги для розвитку своїх додатків, щоб розширити сферу застосування та вплив засобів масової інформації. Отже, видавці відповіли, що вони сформували спільну стратегію щодо популярного принципу «mobile first» (спочатку мобільні пристрої), тому що коли не тільки розвиток, але й дизайн статті орієнтований на мобільну аудиторію, і лише тоді на аудиторію знайомих комп'ютерів.

Такі відмінності пов'язані з контрастними підходами ЗМІ до публікації новин для мобільних телефонів. З одного боку, публікації новин, такі як The Guardian, зосереджуються на публікації адаптованого та важливого контенту, такого як термінові та останні новини плюс загальні бюлетені новин, в той час як The San Francisco Chronicle надає пріоритет матеріалам думок та блогам, оскільки мобільні користувачі набагато активніше, ніж користувачі комп'ютерів.

Існує також певна ніша тематичних програм, які зосереджуються на поширенні інформації на вузьку тему, наприклад, додаток The New York Times'election (заява на вибори New York Times) або додаток для баскетбольної команди Chicago Bulls Chicago [5], а також служби підтримки, такі як путівник ресторану Göteborgs-Posten та шведська гра судоку від Svenska Dagbladet. З іншого боку, деякі компанії, такі як Financial Times, сприймають мобільну платформу дуже скептично, навіть агностично, і взагалі не публікують свої публікації через рідні програми.

Натомість вони використовували "чуйний" або адаптивний веб-дизайн (HTML5), як і інші компанії (Sanoma, the Chicago Tribune и Deseret News).

Інші компанії, такі як американська компанія Digital First Media, стверджують, що їх стратегія не включає ні використання нативного додатка, ні адаптивний веб-дизайн, але поєднує в собі обидва, а також особисте повідомлення про новини через SMS [8]. «Чуйний» веб-дизайн передбачає автоматичну адаптацію вмісту сайтів на будь-якому екрані, будь то мобільний пристрій, комп'ютер, планшет чи телевізор. Основна суть цього методу полягає у тому, щоб уникнути втрати будь-яких пристроїв його потенційних користувачів. За допомогою цього рішення різниця в операційній системі втрачає свою цінність, тому створюється її власна невелика екосистема, що робить вміст більш доступним та керованим [9]. Вони стверджують, що подібний підхід до рідних програм є тим, що вони мають більше обмежень, коли мова йде про гіперпосилання та обмін за допомогою соціальних медіа, таких як Facebook та це не лише недолік у виробництві трафіку користувачів, але й з точки зору бізнесу щодо вимірювання трафіку та перенаправлення користувачів на інші сайти через рекламу.

На закінчення варто зазначити, що все більше великих медіа не покладаються на традиційний унікальний журналістський контент, створений спеціально для мобільних пристроїв. Натомість вони все більше схильні використовувати механізми автоматизованого перепрофілювання журналістського контенту, зосереджуючись на функціональному вдосконаленні процесу читання новин, покращуючи можливість подання контенту.

1.3. Найпопулярніші мобільні операційні системи та їх характеристика

За даними аналітиків на порталі NetMarketShare[11], програмні продукти Apple та Google залишаються основними гравцями на ринку мобільних операційних систем. Щомісяця Windows Phone втрачає десяту частину відсотків продажі. За деякими прогнозами, ОС Microsoft може повністю покинути ринок через те, що не тільки користувачі, але й розробники мобільних додатків втрачають інтерес до нього (Рис 1.1).

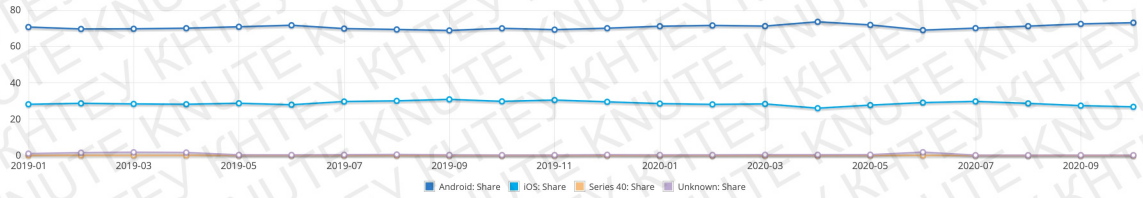

Рис 1.1 статистка платформ на мобільному ринку

Android безкоштовний та відкритий проект. Більшість вихідних кодів поширюється за безкоштовною ліцензією Apache 2.0. Проект почав розвиватися в 2003 році, і вже в 2005 році компанія-розробник була придбана компанією Google. Перша стабільна версія з'явилася у 2008 році [12] Android базується на ядрі Linux, але значно відрізняється від більшості інших систем Linux.

Як і в інших системах Linux, ядро Linux забезпечує такі речі низького рівня, як управління пам'яттю, захист даних, підтримка багатопроцесорного та багатопотокового читання. Але - за кількома винятками - ви не знайдете інших знайомих компонентів систем GNU / Linux в Android: нічого з проекту GNU не використовується, навіть X.Org.

Усі ці компоненти замінюються аналогами, більш пристосованими для використання в умовах обмеженої пам'яті, низької швидкості процесора та мінімального споживання енергії - таким чином, Android більше схожий на вбудовану систему Linux.

Крім того, саме ядро Linux в Android також трохи модифіковане: додано кілька невеликих компонентів, серед яких ashmem (anonymous shared memory), Binder driver (частина фреймворка Binder), wakelocks (управління режимом сна) та low memory killer.

Як libc (стандартна бібліотека мови C), Android не використовує бібліотеку GNU C (glibc), а власну мінімалістичну реалізацію під назвою bionic, оптимізовану для вбудованих систем - це набагато швидше, і менш вимогливо до пам'яті, ніж glibc, яка набула багатьох шарів сумісності(Рис 1.2).

Рис 1.2 android Framework

Багато операційних систем поділяються на саму операційну систему та додатки, встановлені зверху, які нічого не знають один про одного і не знають, як взаємодіяти. Система компонентів та Android з наміром дозволяє програмам, які ще абсолютно нічого не знають один про одного, скласти для користувача один інтегрований досвід роботи - встановлені програми реалізують частини однієї великої системи, вони складають систему. І це, з одного боку, відбувається прозоро для користувача, з іншого - представляє необмежені можливості для налаштування [13.]

Однією з унікальних особливостей Android є те, що програми не безпосередньо контролюють процес, в якому вони запускаються.

Основним блоком у Unix-подібних системах є процес. Як системні послуги низького рівня, так і окремі команди в shell, а графічні програми - це процеси. У більшості випадків процес є чорною скринькою для решти системи - інші компоненти системи не знають і не дбають про її стан. Процес починається з виклику основної main() функції (фактично _start), а потім реалізує частину своєї логіки, взаємодіючи з рештою системи за допомогою системних викликів та найпростішого міжпроцесорного зв'язку(IPC).

Оскільки Android також схожий на Unix, все це справедливо для нього, але хоча деталі низького рівня - на рівні Unix - працюють на концепції процесу, на більш високому рівні - на рівні Android Framework - основним блоком є додаток. Додаток не є чорною скринькою: він складається з окремих компонентів, добре відомих решті системи.

Програми Android не мають основної main() функції, немає жодної точки входу. Загалом, Android максимізує концепцію програми, що працює як від користувача, так і від розробника. Звичайно, процес подання заявки потрібно розпочати і припинити, але Android робить це автоматично. Розробнику пропонується реалізувати кілька окремих компонентів, кожен з яких має свій життєвий цикл.

Для реалізації взаємозв'язку між компонентами необхідно, щоб програми мали можливість спілкуватися один з одним та з системними послугами - іншими словами, потрібен дуже просунутий та швидкий механізм. Для цього використовується механізм Binder..

Binder це платформа для швидкої, зручної та об'єктно-орієнтованої взаємодії між процесами. Binder використовує свій невеликий модуль ядра, взаємодія з яким з userspace користувачів відбувається через системні виклики (переважно ioctl) на "віртуальному пристрої" /dev/binder.

Частини низького рівня Binder працюють з точки зору об'єктів, які можна пересилати між процесами. У цьому випадку кількість посилань (reference-counting) використовується для автоматичного випуску невикористаних спільних ресурсів та повідомлення про посилання на смерть до вільних ресурсів у процесі.

Частини Binder високого рівня працюють з точки зору інтерфейсів, служб та об'єктів проксі. Опис інтерфейсу, наданого програмою іншим програмам, записується спеціальною мовою AIDL (Android Interface Definition Language), яка зовні дуже схожа на оголошення інтерфейсів в Java. Цей опис автоматично генерує справжній інтерфейс Java, який потім можуть використовувати як клієнти, так і сама послуга. Крім того, aidl- файл автоматично генерується двома спеціальними класами: Proxy (для використання клієнтом) та Stub (з сервісу), які реалізують цей інтерфейс.

Для того, щоб різні процеси знайшли послуги один одного, Android має спеціальну послугу ServiceManager, яка зберігає, реєструє та видає жетони всіх інших служб.

Binder широко використовується в Android для реалізації системних сервісів (наприклад, пакетного менеджера і буфера обміну), але деталі цього приховані від розробника додатків високорівневими класами в Android Framework, такими як Activity, Intent і Context. Додатки можуть також

використовувати Binder для надання один одному власних сервісів - наприклад, додаток Google Play Services взагалі не має власного графічного інтерфейсу для користувача, але надає розробникам інших програм можливість користуватися сервісами Google Play.

Основний вид компонентів додатків під Android - це activity. Activity - це один «екран» додатка. Наприклад, в додатку для електронної пошти (email client) можуть бути такі activity, як Inbox Activity (список вхідних листів), Email Activity (читання одного листа), Compose Activity (написання листа) і Settings Activity (настройки).

Передача даних між частинами програми здійснюється за рахунок Intent. Intent - це повідомлення, яке вказує системі, що потрібно «зробити» (наприклад, відкрити даний URL, написати лист на цю адресу, зателефонувати на даний номер телефону або зробити фотографію).

Додатків може знадобитися виконувати дії, не пов'язані безпосередньо з жодною activity, в тому числі, продовжувати робити їх у фоновому режимі, коли всі activity цього додатка завершені. Наприклад, додаток може завантажувати з мережі великий файл, обробляти фотографії, відтворювати музику, синхронізувати дані або просто підтримувати TCP-з'єднання з сервером для отримання повідомлень.

Таку функціональність можна реалізовувати, просто запускаючи окремий потік - це було б для системи чорним ящиком; в тому числі, процес був би завершений при завершенні всіх activity, незалежно від стану таких фонових операцій. Замість цього Android пропонує використовувати ще один вид компонентів - сервіс.

Сервіс потрібен, щоб повідомити системі, що в процесі додатки виконуються дії, які не є частиною activity цього додатка. Сам по собі сервіс не означає створення окремого потоку або процесу - його точки входу (entry points) запускаються в основному потоці додатки. Зазвичай реалізація сервісу

запускає додаткові потоки і управляє ними самостійно.

Окрім activity та послуг, програми Android мають ще два типи компонентів - це broadcast receiver та content provider.

Broadcast receiver - компонент, що дозволяє програмі отримувати broadcast, спеціальний тип повідомлення від системи чи інших програм. В основному broadcas, як випливає з назви, в основному використовувалася для надсилання широкомовних повідомлень до всіх підписаних програм - наприклад, система надсилає повідомлення AIRPLANE_MODE_CHANGED при включенні або вимкненні режиму літака.

Content provider компонент, який дозволяє додатку надавати іншим програмам доступ до даних, якими він керує. Прикладом даних, до яких можна отримати доступ таким чином, є список контактів користувача.

У цьому випадку програма може зберігати самі дані будь-яким чином, у тому числі на пристрої у вигляді файлів, у цій базі даних (SQLite) або запитувати їх у сервера через мережу. У цьому сенсі content provider - це єдиний інтерфейс для доступу до даних, незалежно від форми їх зберігання.

iOS являє собою операційну систему, яка працює на пристроях iPhone, iPad і iPod touch. Операційна система керує апаратним забезпеченням і надає інструменти для розробки додатків. Операційна система iOS також включає різні системні програми, такі як Телефон, Mail, Safari, які є стандартними системними службами для користувача.

Комплект засобів розробки (SDK від англ. Software Development Kit) програмного забезпечення для операційної системи iOS містить інструменти та інтерфейси, необхідні для розробки, установки, запуску і тестування мобільних додатків, які з'являються на домашньому екрані iOS-пристрої. Додатки розробляються з використанням системних бібліотек iOS на мові Objective-C і запускаються безпосередньо на iOS. На відміну від веб-додатків, рідні додатки встановлюються на пристрій і тому завжди доступні

користувачеві, навіть якщо пристрій знаходиться в режимі польоту.

В кінцевому підсумку, iOS виступає в якості посередника між низькорівневим апаратним забезпеченням і додатками, які створюють розробники. Додатки не взаємодіють безпосередньо з апаратним забезпеченням, вони спілкуються з обладнанням через набір чітко визначених системних інтерфейсів. Ці інтерфейси дозволяють легко створювати додатки, які працюють однаково на пристроях, що мають різні апаратні можливості. iOS-технології можна розглядати як набір шарів (Рис 1.3).

Нижні шари містять фундаментальні сервіси та технології. Шари вищого рівня спираються на нижні шари і забезпечують більш досконалі сервіси та технології.

- Cocoa Touch. Шар Cocoa Touch містить ключові фреймворки для створення iOS-додатків. Ці фреймворки визначають зовнішній вигляд додатків і забезпечують базову інфраструктуру додатків і підтримку ключових технологій, таких як багатозадачність, дотики, а push-повідомлення та інших високорівневих системних сервісів

- Media. Шар Media включає в себе графічні, аудіо- та відеотехнологій, які використовуються для реалізації мультимедійних можливостей в додатку.

- Core Services. Шар Core Services містить основні системні сервіси. Ключовими серед них є фреймворки Core Foundation і Foundation, що визначають основні типи, які використовують всі програми. Цей шар також включає в себе специфічні технології для підтримки таких функцій, як геолокація, iCloud, мережеві технології та інші.

- Core OS. Шар Core OS складається з низькорівневих функцій, на яких базується більшість інших технологій. Навіть якщо розробники не використовують ці функції безпосередньо, вони використовуються іншими фреймворками. У ситуаціях, коли необхідно мати справу з безпекою або

зв'язком з апаратним забезпеченням пристрою, можна скористатися фреймворками цього шару.

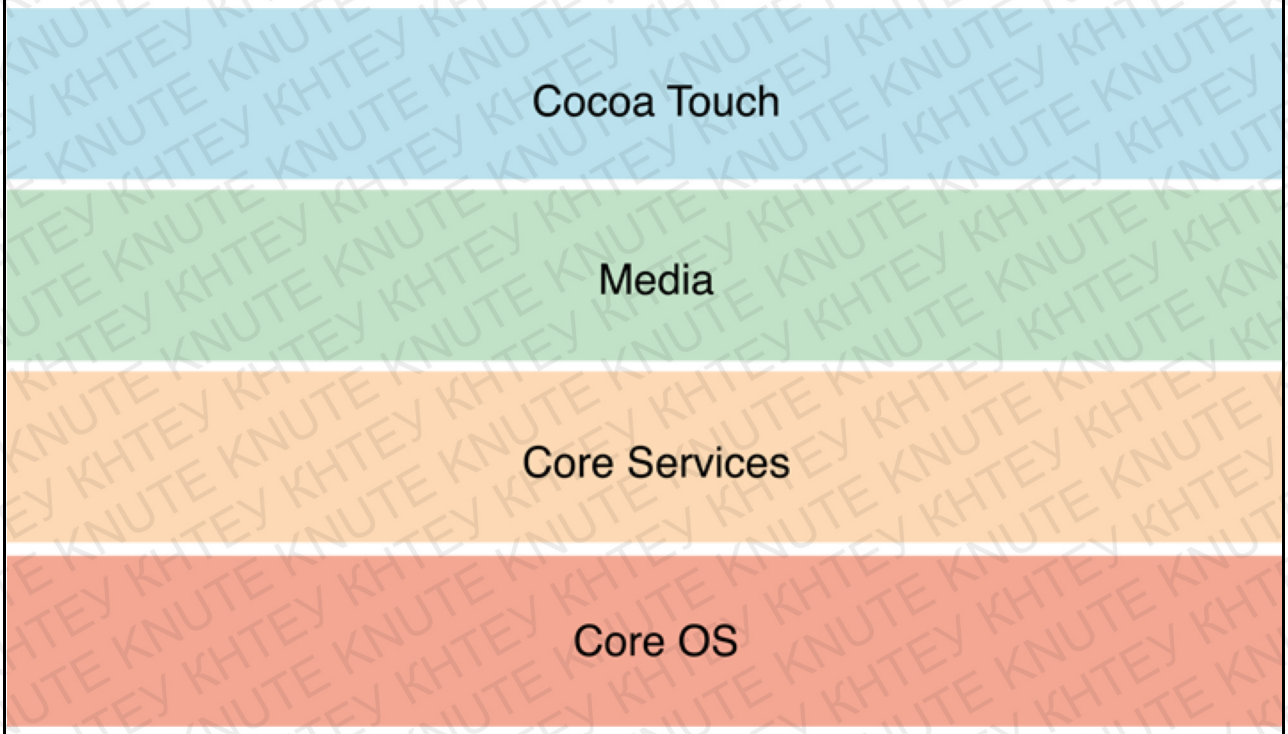

Рис 1.3 шари рівнів iOS

В процесі написання коду слід віддавати перевагу використанню фреймворків більш високого рівня, коли це можливо. Фреймворки вищого рівня надають об'єктно орієнтовані абстракції для конструкцій нижчого рівня. Використання фреймворків низького рівня допускається, якщо вони містять компоненти, недоступні у фреймворками більш високого рівня.

Apple поставляє велику частину своїх системних інтерфейсів в спеціальних пакетах, названих фреймворками. Фреймворк - це директорія, яка містить динамічну бібліотеку загального користування та ресурси (заголовки, зображення, допоміжні додатки), необхідні для підтримки цієї бібліотеки.

1.4. Технічне завдання до розробки додатку

1. Загальні вимоги

Необхідно створити простий та зрозумілий за інтерфейсом мобільний додаток, який надає можливості керувати вихованням дітей.

1.1 Вимоги до інструментів розробки

1.1.1. Інструменти для розробки серверної частини

- мова програмування PHP;

1.1.2. . Інструменти для розробки клієнтської частини

- мова програмування Swift;
- бібліотеки для розробки інтерфейсів UIKit, Foundation, AVPlayer;

2. Вимоги до розробки серверного забезпечення.

2.1. Вимоги до структури.

Серверне забезпечення має дотримуватися послідовної структури, а саме:

- controllers – функції які використовуються в роутері;
- routes – функції для обробки REST-запитів.

3. Вимоги до розробки клієнтської частини.

3.1. Вимоги до оформлення інтерфейсу.

Візуалізація повинна забезпечувати зручний для користувача інтерфейс, який відповідає наступним вимогам:

- наявність локалізованого інтерфейсу користувача;

меню навігації.

1.5. Висновки до розділу 1

Отже мобільні додатки з кожним роком стають все популярніші через більш зручне користування. Операційна система iOS є однією з найпопулярніших сисем у світі. Комплект засобів розробки програмного забезпечення для операційної системи iOS містить інструменти та інтерфейси, необхідні для розробки додатків.

РОЗДІЛ 2. ПОТРЕБА ЗАКЛАДУ ДОШКІЛЬНОЇ ОСВІТИ У СТВОРЕННІ МОБІЛЬНОГО ДОДАТКУ

2.1. Стан закладів дошкільної освіти в Україні

Дошкільна освіта є обов'язковою складовою частиною системи освіти в України. Її функціонування регламентоване Конституцією України, законами України «Про освіту», «Про дошкільну освіту», «Про охорону дитинства», конвенцією ООН про права дитини.

Дошкільний навчальний заклад — це навчальний заклад, який забезпечує реалізацію права дитини на дошкільну освіту, його фізичний, психічний та духовний розвиток, соціальну адаптацію та готовність продовжувати освіту.

Призначення полягає у тому щоб задовольнити потреби громадян в здобутті дошкільної освіти, безпечні та нешкідливі умови розвитку, виховання та навчання дітей, зформувати у дітей гігієнічні навички та основи здорового способу життя, сприяти збереженню та зміцненню здоров'я, розумовому, психологічному і фізичному розвитку дітей. [22] (Рис 2.1)

Станом на 5 листопада 2020 року в навчальних закладах для дітей дошкільного віку було створено 10 282 додаткові місця. Міністерство освіти і науки отримало відповідні оперативні дані від регіональних управлінь (урядів) освіти та науки. «Деякі густонаселені регіони досі стикаються з проблемою нестачі місць у дитсадках, тож ми продовжуємо координувати діяльність місцевих органів виконавчої влади щодо відкриття й реконструкції закладів дошкільної освіти, а також створення додаткових груп.

Мережа та контингент закладів дошкільної освіти на кінець 2016 року[3]

	Кількість закладів, од			В них місць, од			Кількість дітей, осіб		
	усього	у тому числі		усього	у тому числі		усього	у тому числі	
		міські поселення	сільська місцевість		міські поселення	сільська місцевість		міські поселення	сільська місцевість
Усього	**14949**	**5712**	**9237**	**1124909**	**785181**	**339728**	**1300129**	**983876**	**316253**
у тому числі									
дитячі садки	4856	578	4278	185254	49946	135308	182239	58921	123318
ясла-садки	6772	4632	2140	815119	687839	127280	994353	871257	123096
навчально-виховні комплекси	3321	502	2819	124536	47396	77140	123537	53698	69839
Із загальної кількості дошкільні заклади									
загального розвитку	13101	3912	9189	814692	478508	336184	918107	605691	312416
комбіновані	1565	1521	44	276017	272710	3307	344376	340850	3526
санаторні	134	131	3	18589	18422	167	22395	22122	273
спеціальні	149	148	1	15611	15541	70	15251	15213	38

Рис 2.1 - статистика закладів дошкільної освіти України

Найпроблемнішим питанням залишається недостатня кількість місць у садочках. Зараз в Україні діє близько 15 тисяч закладів дошкільної освіти, а на черзі перебувають приблизно 25 тисяч дітей.

У цілому по Україні в сільській місцевості знаходиться 9 тисяч садочків, а у містах — 6. І найбільші черги у містах. Ця проблема створена через постійну міграцію.

Від проблеми з недостатньої кількості місць у дитячих садочках страждає кожне місто в Україні. Нові будівельні норми, передбачені документом ДБН В.2.2-4:2018 «Будинки і споруди: заклади дошкільної освіти», що вступили в дію з 1 жовтня 2018 року, дозволили проектування вбудованих садочків. Уже сьогодні будівельні та девелоперські компанії активно продукують нові проекти, а також вносять зміни в уже існуючі, проектуючи вбудовані садочки. Для вирішення цієї проблеми батьки вирішили створювати приватні садочки. Поліція часто викриває незаконні приватні садочоки які не мають 6 м2 на 1 дитину, або не мают ліцензії кваліфікують як незаконні.

Додаткові місця з початку 2020 року було створено шляхом: відкриття нових закладів - 16 новобудов, 1 495 місць; відкриття приватних дитячих садків – 96 закладів, 3 626 місць; проведення реконструкції нинішніх дитсадків – 35 закладів, 2 557 місць; відкриття дитячих садків у пристосованих приміщеннях – 10 закладів, 266 місць; відкриття додаткових груп у дитсадках; що функціонують – 85 груп; 1 685 місць; відкриття груп із короткотривалим перебуванням дітей – 11 груп; 158 місць. Найбільше додаткових місць для дітей дошкільного віку у 2020 році: Київська область – 2 585 місць; м. Київ – 2 000 місць; Донецька область – 802 місця; Харківська область – 699 місць.

Але бувають випадки, коли власники та батьки, які не мають іншого вибору, як мати такі дитячі садки, трактують це так: "Ми, сусіди, зібралися і зробили дитячий садок сімейного типу. У нас немає ліцензії, але це не дитячий садок - тому ми разом піклуємося про своїх дітей Доки є проблема з місцями в садках і немає регулювання норм для приватного сектору, доти і виникатимуть такі ситуації, тож це треба врегулювати. [23]

2.2. Вирішення проблем батьків та приватних закладів дошкільної освіти

Щомісячна вартість життя в приватному дитячому садку в столиці становить від 7 до 22 тисяч. Осклільки середня заробітна плата в Україні складає 12 тисяч це насправді дорого, але і утримувати дитячий садок недешево[5]. Садок мусить сплачувати оренду, комунальні послуги, заробітну плату, купувати продукти харчування або оплачувати рахунки компаніям, які постачають їжу, обладнання, меблі, іграшки, канцтовари та матеріали для розвитку дитини. У травні 2016 року набули чинності нові санітарні правила для дошкільних навчальних закладів. Через зміни багатьом приватним садам вдалося працювати легально, адже зробити роботу приватного дитячого садка повністю законною, можливості майже не було. Таким чином, відповідно до

нових санітарних стандартів, дитячі садки можуть розташовуватися в квартирах, немає необхідності мати власний харчовий блок, можна використовувати замовлення їжі, але так щоб вона відповідала всім стандартам.[6]

Але у праватних садочках існує дужа багато проблем. Які виникають зазвичай через недостатню кількість кадрів або фінансування. Щодо кадрів не завжди у приватному садочку з вашою дитиною буде працювати кваліфікований спеціаліст.

У Києві стався показовий випадок: перехожі знайшли хлопчика у парку Наталка на Оболоні. Малюк не міг розказати, де його батьки та з ким він прийшов, тому небайдужі викликали патрульну поліцію. Дві (!) години поліція шукала батьків дитини, обдзвонювала усі приватні та державні заклади. А престижний садочок району (з безліччю гарних відгуків!), який "загубив" малюка на прогулянці, його навіть не шукав. Вже після дзвінка поліції, прорахувавши дітей, вони визнали, що це їхній вихованець. Але уявіть, що могло статися з малюком через неуважність та халатність вихователів.[4]

Однорічна дівчинка, яку забрала швидка з нелегального садочку у Запоріжжі, не померла власною смертю, вихователька прив'язала їй до рота подушку і дитина захлинулась від блювоти.[7] Також існуют дуже багато випадків з отруенням дітей, коли в дитини є непереносимість лактозі, або інших продуктів. Випадки коли дитина має фізічні вади та не може виконувати вправи, або просто перенапружується.

Вирішити ці проблеми допоможе віддалений звязок. Якщо батьки матимуть змогу контролювати процесс на відстані то всі форсмажорні випадки можуть бути уникнути. Наприклад якщо батько або мати побачать в меню страв продукт, який дитині не можна споживати вони зможуть про це повідомити. В розкладі занять побачити фіз-культуру, попередити про якісь

фізичні вади. Та також мати змогу дистанційно переглядати завдяки камерам спостереження, що відбувается з їх дитиною. Для європейського ринку вже існує таке рішення. Проект Baby's Days[8] - це найбільш усталений і найсучасніший програмний пакет у галузі догляду за дітьми, який розвивається в безпечній системі з квітня 2009 року для широкого спектру постачальників послуг з догляду за дітьми, починаючи від дитячих садків, до ясел та дошкільних закладів.

Додаток Baby's Days тепер ще більше полегшує життя клієнтам та батькам Baby's Days, запуск нашого складного додатка також пришвидшить ваш робочий день. Окрім відстеження навчання та розвитку кожної дитини за допомогою спостережень, наступних кроків, COEL, відео та фотографій, ви також можете виконувати щоденні роботи безпосередньо з програми лише за кілька кліків.

Baby's Days - це не просто ще одне програмне забезпечення для створення навчального журналу шляхом додавання спостережень та наступних кроків для дітей, це програмне забезпечення, яке дозволяє персоналу швидко та ефективно виконувати необхідні документи в рамках догляду за дітьми та включає більше функцій, ніж будь-яка інша система на ринку, подивіться на меню праворуч, щоб побачити всі функції, які містять Baby's Days:

Система Baby's Days - це єдине програмне забезпечення такого типу, яке доступне в будь-якій точці світу для розплідників та дітей, і забезпечує програмне забезпечення для зупинки для персоналу розплідника та дітей.

Доведено, що інтелектуальна система економить ваші години на папері, трудомісткі завдання, такі як запис пляшок дитини до моніторингу прогресу дитини зараз займає всього кілька хвилин, не потрібно вводити необхідні дані більше одного разу як всі області системи спілкуються між собою.

Можна вести облік годування, заміну підгузників, час сну або більше. Батьки можуть додавати спостереження, наступні кроки для розвитку, завантажувати фотографії, записувати нещасні випадки та інциденти, що сталися вдома, записувати ліки які дитина приймае, надсилати приватні повідомлення, залишати нотатки та багато іншого.

2.3. Огляд потрібного обладнання

Основним обладнанням для системи відеоспостереження є камери відеоспостереження та пристроїв обробки та запису. Камери спостереження є основним і основним компонентом будь-якої системи відеоспостереження, вони просто знімають зображення, яке передається далі на монітор і на пристрій запису. Їх можна поділити на декілька характеристик: колір(кольорова або чорнобіла), чутливість(здатність камери відображати погано освітлені обьєкти), ТВЛ (кількість вертикальних ТВ ліній, чим більше, тим дозвіл картинки більше), кут огляду камери.

Важливий момент при побудові системи відеоспостереження - точно визначити цілі її установки і завдання, які повинна вирішувати система відеоспостереження, після цього замовнику і проектувальнику значно легше визначитися з критеріями якості та підібрати обладнання для системи відеоспостереження. Природно, при розрахунку системи відеоспостереження необхідно технічне обстеження об'єкта та складання проекту.

Нові системи відеоспостереження будуються із застосуванням IP камер і цифрових систем обробки (NVR) і / або серверів і відповідного програмного забезпечення. Як пристрої для перегляду відеоінформації можна використовувати смартфони та планшети, ноутбуки, традиційні комп'ютери, також широко використовуються хмарні технології.

Для плавного переходу від застарілих систем відеоспостереження із застосуванням аналогових відеокамер використовуються технології HD відеоспостереження (HD TVI або Turbo HD, HDCVI, AHD), які можуть по існуючих комунікацій (коаксіальному кабелю) істотно підняти якість відеосистеми (перейти від 0,3MP до 2-8MP).

Досить часто із застосуванням HD відеокамер і HD DVR будують нову систему відеоспостереження - вона в багатьох випадках трохи дешевше, ніж повністю цифрова система на базі IP технологій.

Устаткування системи відеоспостереження незалежно від виду системи (аналогова, гібридна або цифрова) складається з відеокамер (забезпечують збір відеоінформації), пристроїв обробки та запису - DVR, NVR, плати відеозахвату, сервер з прикладним ПО, архівуючого обладнання та пристроїв відображення відеоінформації (монітор).

Пристрої обробки та запису - відеореєстратори (DVR, NVR), відеосервери - дозволяють обробляти інформацію, отримані з відеокамер, аналізувати відеопотоки, записувати їх та публікувати їх у заданому форматі на моніторі та в локальній чи глобальній мережі для віддаленого перегляду. Вони також дозволяють записувати зображення як безпосередньо з відеокамери, так і з пристроїв обробки. Сучасні системи відеоспостереження зазвичай використовують звичайні жорсткі магнітні диски для зберігання та зберігання відеоінформації (HDD), а при великому обсязі інформації - накопичувачі, об'єднані в RAID.

Комп'ютерні монітори зараз використовуються як пристрої відображення інформації для систем відеоспостереження, також відеоінформація переглядається на моніторі комп'ютера, мобільному телефоні за допомогою програми або веб-браузера. Особливою популярністю користуються програми для відеоспостереження для Android та iOS.

Дуже важливою складовою будь-якої системи відеоспостереження є середовище передачі сигналу - кабельна мережа та джерело живлення. Створюючи систему відеоспостереження за допомогою IP-технологій, кабельна система базується на принципах локальних комп'ютерних мереж (LAN) і може використовувати інше фізичне середовище для передачі інформації - мідний кабель, такий як «вита пара", оптичний кабель, WiFi. Для живлення відеокамер використовується силовий кабель, але останнім часом набирають популярність технології POE (power ower ethernet), де живлення і дані передаються по одному кабелю типу вита пара.

Найчастіше якість компонентів та матеріалів кабельної мережі та джерела живлення залежить від якості системи відеоспостереження в цілому, оскільки кінцеві пристрої - відеокамери, відеореєстратори, мають дуже хороші параметри. Насправді, передаючи відеосигнал на відстані, існують інженерні завдання для створення високоякісної системи відеоспостереження, тобто на передньому плані розробки та впровадження системи відеоспостереження - вибір носія передачі, компетентний розрахунок кабельної або бездротової мережі з урахуванням постійно зростаючої роздільної здатності відеокамер та пристроїв відеоспостереження.

2.4 Висновки до розділу 2

Найпроблемнішим питанням залишається недостатня кількість місць у садочках. Для вирішення цієї проблеми батьки вирішили створювати приватні садочки. Але часто такі садочки є нелегальними. Вирішити проблеми з контролем та вихованням дітей у приватних садочків зможе мобільний додаток, та додаткова техніка для відео спостереження.

РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ОПЕРАЦІЙНОЇ СИСТЕМИ IOS

3.1. Засоби розробки

Xcode – це пакет інструментів для розробки додатків під Mac OS X, розроблений Apple. Xcode безкоштовно можно завантажити з Appstore. Оновлення можна безкоштовно скачати на офіційному сайті підтримки. На сьогодні є єдиним засобом написання «універсальних» (Universal Binary) програм для Mac OS та iOS.

Xcode інтегрований з фреймворком Cocoa. Його використовують і при розробці самої Mac OS . Цей набір інструментів включає:

Xcode IDE (для кодування, створення і налагодження додатків) (Рис. 3.1)

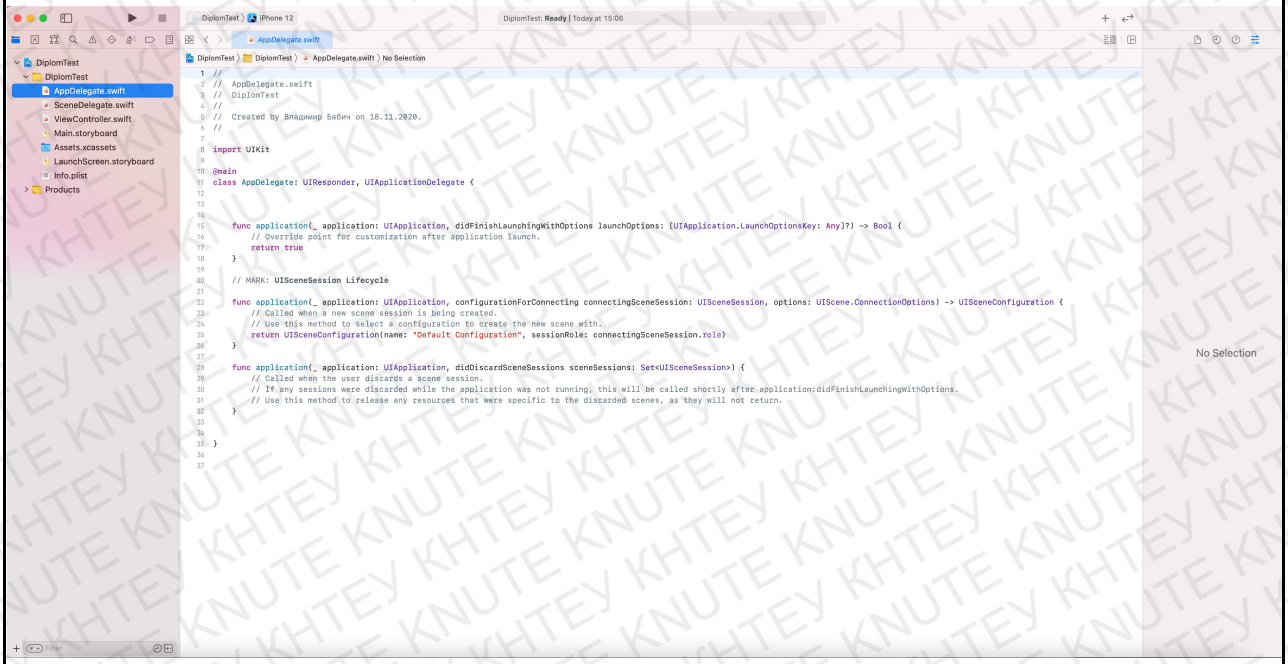

Рис 3.1 Xcode IDE

					КНТЕУ 121 06з-1.МР			
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*				
Зав. каф.		Криворучко О.В.		19.10.20	«Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»	Стадія	Аркуш	Аркушів
Керівник		Цензура М.А.		19.10.20		РЗ	36	78
Гарант		Криворучко О.В.		19.10.20		*Факультет інформаційних технологій 2м курс, 6з група*		
Розробив		Бабич В.О.		19.10.20	*Розробка мобільного додатку для операційної системи IOS*			

Xcode IDE надає все, що потрібно для розробки: редакторы з функцією автозаповнення коду та Cocoa рефакторінга, до налаштування open-source компіляторів. Xcode IDE розроблений з нуля, щоб можна було скористатися всіма можливостями Cocoa і новітніми технологіями Apple.

Xcode включає велику кількість можливостей, щоб полегшити розробки iOS-додатків, включаючи наступні:

- систему управління проектом для визначення програмних продуктів;
- середовище для редагування коду, що включає можливості підсвічування синтаксису, автозаповнення коду та індексацію символів;
- просунуту програму перегляду і пошуку документації Apple;
контекстно-залежний інспектор для перегляду інформації про виділений в коді символі;
- просунуту систему збирання проекту з перевіркою залежностей і перевіркою правил складання;
- компілятори GCC, що підтримують мови C, C ++, Objective-C, Objective-C ++ та інші;
- підтримка LLVM і Clang для мов C, C ++ і Objective-C;
- вбудоване налагодження вихідного коду з використанням GDB;
- розподілені обчислення, що дозволяють поширювати великі
- проекти через кілька мережевих пристроїв;
- інтелектуальна компіляція, яка прискорює час компіляції одного файлу;
- просунуті можливості по налагодженню коду;
- просунуті засоби налагодження, що дозволяють робити глобальні
- зміни в коді без зміни його поведінки;
- підтримка знімків проекту, які надають легше управління вихідним кодом;
- підтримка запуску засобів продуктивності для аналізу програми;

- підтримка вбудованої системи управління вихідним кодом;

- підтримка AppleScript для автоматизації процесу складання;

- підтримка налагоджувальної інформації в форматах DWARF і Stabs (налагоджувальна інформація для всіх проектів за замовчуванням генерується в форматі DWARF).

- Interface Builder (для розробки користувальницького інтерфейсу) (рис. 3.2) [18]

Рис 3.2 Interface Builder

Редактор Interface Builder в Xcode спрощує розробку повноцінного інтерфейсу користувача без написання коду. Просто перетягніть вікна, кнопки, текстові поля та інші об'єкти на полотно дизайну, щоб створити функціонуючий користувальницький інтерфейс.

Оскільки Cocoa та Cocoa Touch побудовані з використанням шаблону Model-View-Controller, легко розробляти інтерфейси самостійно, окремо від їх реалізації. Користувальницькі інтерфейси - це фактично заархівовані об'єкти Cocoa або Cocoa Touch (збережені як файли .nib), а macOS та iOS динамічно створюватимуть зв'язок між інтерфейсом користувача та кодом під час запуску програми.

Використовуючи Interface Builder, можна створювати вікна програми шляхом перетягування в нього попередньо налаштованих компонентів. Компоненти включають стандартні системні елементи управління, такі як перемикачі, текстові поля, кнопки тощо. Компоненти переміщуються у вікна, після цього їх можна позиціонувати, перетягуючи по всій величині вікна, налаштовувати атрибути, використовуючи інспектор і встановлювати зв'язки між цими об'єктами і кодом. Вміст вікна зберігається в nib-файл, який є ресурсним файлом особливого формату. Він містить всю інформацію, яка потрібна бібліотеці "UIKit" для створення тих же самих об'єктів у додатку під час виконання. Завантаження nib-файлу створює версії часу виконання всіх об'єктів, збережених у файлі, налаштовує їх в точності, якими вони були в Interface Builder. Також використовується інформація про взаємозв'язок, зазначена розробником для установки зв'язку між заново створеними об'єктами і вже існуючими в додатку. Ці зв'язки забезпечують код вказівниками на об'єкти nib-файлу, а також надають інформацію самим об'єктам, необхідну для передачі дій користувача вихідному коду.[19]

Використання Interface Builder зберігає час, що припадає на створення UI. Interface Builder усуває написання коду, необхідного для створення, налаштування і позиціонування об'єктів, які використовуються для розробки інтерфейсу. UI фактично є архівами об'єктів Cocoa, які не вимагають генерації коду. Зміни в інтерфейсі користувача (UI) не вимагають перекомпіляції (перевірки) коду, а зміни в коді не вимагають перекомпіляції UI.

- Інструменти для аналізу поведінки і продуктивності (рис. 3.3)

Рис 3.3 Developer Tools

Developer Tools це інструменти оптимізації і аналізу (Instruments and Shark), які дозволяють проаналізувати продуктивність iOS-додатків, поки вони виконуються в емуляторі або на пристрої. Developer Tools збирають дані запущеного додатку і відображають їх у графічному вигляді (тимчасова шкала). Вона дозволяє збирати дані про використання пам'яті, дискової активності, мережеву активність і графічної продуктивності мобільного застосування. Тимчасова шкала може відображати всі типи інформації відразу, дозволяючи співвідносити загальну поведінку додатка, а не тільки поведінку за певним параметром. Все це дозволяє легко визначити проблемні зони додатка, і потім перейти до проблемних рядків коду.[18] Developer Tools надають засоби для аналізу поведінки програми з плином часу. Наприклад, вікно середовища Developer Tools дозволяє зберігати дані декількох запусків,

- дозволяючи тим самим бачити, чи поліпшилося поведінка застосування або воно все ще потребує доопрацювання. Також можна зберігати дані цих запусків в документах і відкривати їх в будь-який час.

3.2. Огляд мов програмування для розробки

Процес розробки додатку - це написання програми на одній з основних мов iOS-розробки: Objective-C або Swift. Цей шлях створений корпорацією Apple і передбачає дотримання всіх її ідеям. Apple забезпечує розробників останніми версіями SDK (software development kit - набір засобів розробки), документацією, а також середовищем розробки Xcode.

Обидві мови для створення iOS-додатків відносяться до об'єктно-орієнтованого програмування (ООП). Його основні парадигми: спадкування, поліморфізм, інкапсуляцію і абстракцію. Простими словами, ООП - це стиль написання коду, який дозволяє розробнику групувати схожі завдання в класи. Код відповідає принципу DRY (do not repeat yourself - повторюй самого себе) і стає легким для супроводу.

Objective-C. Ветеран з великою історією, поступово відходить на другий план. Мова програмування для iOS додатків, створений на початку 1980-х років минулого століття шляхом схрещування C (Сі) з популярним в той час Smalltalk (зв'язок з об'єктами через повідомлення). Objective-C спочатку сприймався, як проста надбудова над мовою C, що модифікує його деякі синтаксичні конструкції, але після того, як за ліцензування взялася спочатку компанія Next Step, а потім на правах наступника і Apple, Objective-C став одним з найбільш популярних мов для розробки додатків для iPhone і iPad. Тому багато типів даних в Objective-C успадкували префікс NS (Next Step). Це основна мова, що використовується компанією Apple, знання якої дозволяє писати під будь-які платформи Apple, в тому числі macOS [14].

Плюси:

- висока ступінь підтримувані коду: з кожним оновленням зміни в Objective-C мінімальні;
- велика кількість документації, технічної літератури та величезне співтовариство Apple надає і регулярно оновлює офіційні книги іресурси.
- швидкий перехід з однієї з мов сімейства C. Objective-C - це розширення мови C це означає, що будь-який код на C є також коректним кодом і для Objective-C, потрібно тільки звикнути до синтаксису;
- сумісність Objective-C всередині проектів, написаних на Swift, дозволить застосовувати дві мови одночасно.

Мінуси:

- можуть виникнути складності розуміння принципів ООП і нагромаджених синтаксису;
- низька читаність коду: на початку вивчення синтаксис здасться складним;
- динамічна система типів даних, яка також є плюсом, передбачає можливість появи або пропуску помилок навіть під час компіляції. Зокрема, затягнути процес можуть помилки;
- низька в порівнянні з мовою Swift продуктивність;
- взаємодія з файлами Swift відбувається за допомогою «моста» (умовний адаптер, який переводить код на Swift в формат Objective-C), що сильно гальмує процес складання.

Swift дуже молодий і швидко набирає популярність серед розробників. Офіційно представлений компанією Apple 2 червня 2014 року. Поєднує в собі все краще від C і Objective-C, але позбавлений обмежень останнього, що накладаються на догоду сумісності з C. У Swift використовуються сувора типізація об'єктів, що зменшує кількість помилок ще на етапі написання коду. Також в Swift додані сучасні функції, такі як дженерики, замикання, множинні повернені значення і багато іншого, що перетворюють створення додатка в

більш гнучкий і захоплюючий процес. Основою нової мови програмування послужили існуючі компілятор, відладчик і фреймворки [15].

Плюси:

- швидкість. Зараз мова майже на одному рівні з C ++, і в Купертіно обіцяють, що це не межа;
- спрощена навігація по файлах проекту. На відміну від Objective-C, який створює два файли для оголошення і реалізації, Swift обходиться всього одним. Крім того, імена методів і коментарі між файлами синхронізуються автоматично;
- легка читаність, оскільки дана мова не побудована на C. Наприклад, не потрібно ставити крапку з комою в кінці рядка і писати дужки для оточення вираження всередині if / else. Ніяких квадратних дужок, Swift нагадує звичайний англійську мову, є набагато чистішим і має спрощений синтаксис;
- лаконічність. Кількість коду зі Swift стає набагато менше. Наприклад, для додавання двох рядків можна скористатися оператором «+»;
- великі можливості в порівнянні з Objective-C. Наприклад, дженерики (універсальні шаблони). Універсальний код дозволить тобі писати гнучкі, загального призначення, функції та типи, які можуть працювати з будь-якими іншими типами. Можна написати код, який не повторюється і висловлює свій контент в абстрактній формі;
- повна взаємодія з кодом, написаним на Objective-C, дозволить тобі застосовувати дві мови одночасно;
- підвищена безпека. Swift, на відміну від Objective-C, строго типізований, тобто при оголошенні іменованих параметрів потрібно явно вказувати тип даних, інакше при виконанні коду компілятор викликає помилку. Це полегшить процес усунення багів, оскільки ти можеш вирішити проблему відразу;

- підтримка динамічних бібліотек. Одне із значущих змін в Swift - перехід від статичних бібліотек до динамічних, які по суті є виконуваними шматками коду. Вони приєднуються до додатка і «зв'язуються» з новими версіями мови, що дозволяє програмі працювати стабільно.

Мінуси:

- Swift постійно розвивається і змінюється. Наприклад, виклик методу може змінитися після оновлення. Благо Apple збудували цей процес таким чином, що код написаний на більш ранніх версій не буде зламаний. Ти тільки побачиш попередження про те, що твій код написаний на старій версії, і редактор запропонує перехід на більш нову і допоможе виконати цей процес через підрядник;
- взаємодія з файлами Objective-C відбувається за допомогою «моста», який сильно гальмує процес складання.

3.3. Розробка архітектури розробляемого проекту

Патерн проектування Model-View-Controller (MVC) дає об'єктам одну з трьох ролей: модель (model), уявлення (view) або контролер (controller). Цей шаблон не тільки розподіляє ролі у об'єктів, але так само визначає те, як вони будуть взаємодіяти один з одним. Кожен з трьох типів об'єктів відділений від інших абстрактними законами (якщо можна це так назвати) і зв'язується з іншими об'єктами за допомогою цих законів.

MVC займає почесне місце у проектуванні Cocoa додатків. У цього патерну багато переваг. Наприклад багато обьектів у додатку, можно використовувати багато раз в незалежності від інших типів. Додаток MVC краще та простіше розширювати. Багато технологій Cocoa створені з використанням MVC, тому потрібно щоб створені обьєкти виконували ролі MVC(Рис 3.4).

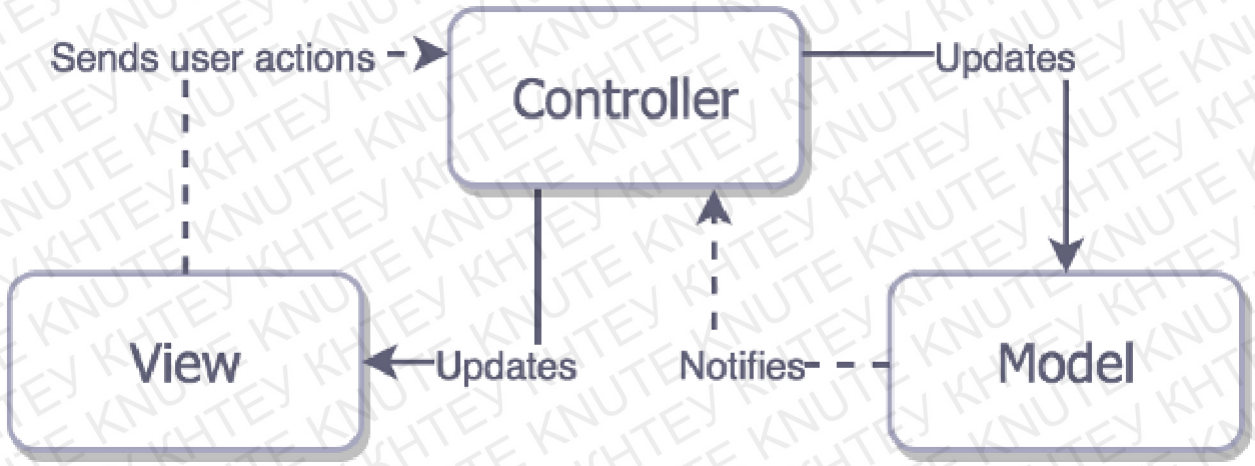

Рисунок 3.4 архітектура MVC

Cocoa MVC заохочує вас писати Massive View Controller, тому що контролер настільки залучений в життєвий цикл View, що важко сказати, що він є окремою сутністю. Хоча у вас все ще є можливість відвантажити частину бізнес-логіки і перетворення даних в Model, коли справа доходить до відвантаження роботи у View, у вас не так багато варіантів. У більшості випадків вся відповідальність View полягає в тому, щоб відправити дії до контролера. У підсумку все закінчується тим, що View Controller стає делегатом і джерелом даних, а також місцем запуску і скасування серверних запитів і, в общем-то, всього чого завгодно.[16]

Об'єкти моделі інкапсулюють дані, специфічні для докладання, і визначають логіку, обчислення, які керують і обробляють ваші дані. Наприклад, об'єктом моделі може представляти із себе персонаж в грі або контакт з адресної книги. Об'єкт моделі може мати відношення «один-ко-многим» з іншими об'єктами моделі, тому іноді модельний рівень додатки є одним або декількома об'єктними графами і мати ієрархічну структуру. Більша частина даних, яка постійно зберігається в додатку, повинна знаходитися в об'єктах моделі після завантаження даних в додаток. Оскільки об'єкти моделі являють собою якісь знання, пов'язані з конкретним завданням, їх можна повторно використовувати в схожих завданнях. В ідеалі, об'єкт моделі не

повинен мати явного зв'язку з об'єктами уявлення, які надають свої дані, і дозволяють користувачам редагувати ці дані - це не повинно стосуватися завдань призначеного для користувача інтерфейсу.

Зв'язок об'єктів: Користувальницькі дії на рівні уявлення (view), які створюють або змінюють дані, повинні передаватися через об'єкт контролера (controller) і приводити до створення або відновлення моделі. Коли об'єкт моделі змінюється (update), то він повідомляє (notify) контролер, який оновлює відповідний view в додатку.

Об'єкти уявлення це такі об'єкти, які користувач бачить на екрані. Ці об'єкти знають як себе показувати і можуть реагувати на дії користувача. Основна мета об'єктів предствлений - відображати дані з об'єктів моделі програми та дозволяти редагування цих даних. Незважаючи на це, об'єкти view зазвичай відділяються від об'єктів model в додатку MVC.

Оскільки ви зазвичай повторно використовуєте і змінюєте дані, view об'єкти забезпечують узгодженість між додатками. Обидві структури UIKit і AppKit надають колекції класів уявлень, а Interface Builder пропонує десятки об'єктів виду в своїй Бібліотеці. Обидві структури UIKit і AppKit надають колекції класів уявлень, а Interface Builder пропонує десятки об'єктів виду в своїй бібліотеці.

Зв'язок об'єктів: view об'єкти дізнаються про зміни в даних моделі через об'єкти контролера додатки і повідомляють про зміни, ініційованих користувачем, наприклад текст, введений в текстові поля дані, проходять через об'єкти контролера об'єктів моделі програми.

Об'єкт контролера виконує роль посередника між одним або декількома view об'єктами докладання і об'єктами model. Об'єкти контролера, таким чином, є каналом, через який view об'єкти дізнаються про зміни в об'єктах моделі і навпаки. Об'єкти контролера також можуть виконувати налаштування і координування завдань для програми та керувати життєвими циклами інших

об’єктів.[17]

Зв'язок об'єктів: Об'єкт контролера інтерпретує дії користувача, створені у view об'єктах, і передає нові або змінені дані на модельний рівень. Коли об'єкти моделі змінюються, об'єкт контролера пов'язує ці нові дані моделі з об'єктами уявлення, щоб вони могли його відображати.

Cocoa MVC - це розумний вибір, якщо ви не готові інвестувати багато часу в свою архітектуру і відчуваєте, що патерн з більш високою вартістю обслуговування вашому невеликому проекту або стартапу буде не по кишені.

Cocoa MVC є найкращим архітектурним патерном з точки зору швидкості розробки.

Тому для розробки буде використано патерн MVC. У додатку для роботи з закладами дошкільної освіти, мусить бути присутня авторизація на сервері. Користувачів (батьків) потрібно відокремети від інших, тому для кожного буде створено унікальні логін та пароль. Після авторизації юзер мусить отримати та мати можливість керувати опціями дітей. Оскільки у батьків дітей може бути декілька спершу оброблюється запит на кількість та ідентифікатори дітей, а після отримання опції на обрану дитину. В опціях повинен бути контент з запису камер - url для перегляду відео та дати коли це відео було створено; url картинки розкладу занять; картинки меню страв.

Проект буде складатись:

- Клас котрий відповідає за роботу з сервером
- Клас якій буде зберігати отриманні данні
- Контролер авторизації
- Контролер з відображенням дітей користувача
- Контролер опцій які притаманні данній дитині
- Контролери які віподображають кожну з отриманних опцій з сервера

3.4. Розробка серверної частини

PHP став однією з найбільш широко використовуваних мов програмування. За даними WWW Technology Surveys, частка його використання у всьому світі перевищує 80%. Серед прихильників PHP є такі гіганти, як Slack, Wikipedia, Wordpress, Pinterest, Nvidia, Tumblr і, певною мірою, Facebook. Широкий спектр функцій, якими він володіє, дозволяє застосовувати цю мову практично в будь-якій галузі розвитку IT. Ось чому компанії - від стартапів до великих підприємств - так часто вибирають цю мову програмування, коли йдеться про розробку продуктів та послуг.

Основна мета мобільного додатка - залучити користувачів або партнерів до вашого бізнесу. Тут персоналізація досвіду та створення правильного контексту є надзвичайно важливими. Ось чому сьогодні мало ділових мобільних додатків є автономними, і більшість із них спілкуються із сервісними службами. Серверна частина програми відповідає за агрегування різних даних з мобільного пристрою, шаблони поведінки користувачів, збереження налаштувань користувача тощо. Декілька фреймворків PHP, таких як Symfony або Laravel, цілком непогані для створення внутрішньої функціональності мобільний додаток.

У поєднанні з кількома іншими методами, заснованими на різних мовах, це призвело до створення добре працюючого продукту.

Завдяки добре налагодженій спільноті, широкому набору функцій і досить великому набору фреймворків, PHP може робити майже все. Збір даних, сценарії на стороні сервера, динамічне генерування вмісту сторінки - це лише декілька областей його використання. PHP можна використовувати у всіх основних операційних системах, включаючи Microsoft Windows, Linux, багато варіантів Unix та macOS. Він також підтримує більшість веб-серверів та баз даних.[20]

Для розробки нашого додатку на сервері мусить бути створено три запити. Перший авторизація, який буде приймати два тектових значення, а повертати результат в якому будут знаходитись параметри дітей.

Приклад документу API :

```
HTTP method: POST
URL: http://mybackend.com/login
BODY:
{
        «uid» : 123,
        «pwd» : "3CB3E2E6AECA48C41000119767B561F5E9E66229" // contains sha1(pass+salt)
        «userName» : «name»
        «userImage» : «https://mybackend.com/images/3CB3E2E6AECA48C41000119767B561F5E9E66229/1.jpg»
        «userChildrens» : [{«Mike» : {«id» : «3CB3E2E6AECA48C41000119767B561F5E9E66229.1», «image» : «https://mybackend.com/images/3CB3E2E6AECA48C41000119767B561F5E9E66229.1/1.jpg» }, «Bob»: {«id» : «3CB3E2E6AECA48C41000119767B561F5E9E66229.2», «image» : «https://mybackend.com/images/3CB3E2E6AECA48C41000119767B561F5E9E66229.2/1.jpg» }}]
 }
```

Другий метод це метод logout, який буде використовуватись для виходу з додатку. Він нічого не приймає, а повертае результат success.

```
HTTP method: GET
URL: http://mybackend.com/logout
BODY:
{
        «success» : «success»
}
```

Третій метод буде приймати ідентифікатор дитини, та повертати доступні опції, та параметри для них.

HTTP method: POST
URL: http://mybackend.com/childOptions
BODY:

```
{
    «uid» : 123,
    «pwd» : "3CB3E2E6AECA48C41000119767B561F5E9E66229" // contains childrenID
    «videoUrl» : «https://mybeckend.com/{childrenID}/video»
    «weekMenuImage» : «https://mybackend.com/images/menu/1.jpg»
    «scheduleImage» : »https://mybackend.com/images/schedule/1.jpg»
}
```

3.5. Розробка клієнтської частини

Створимо проект в IDE Xcode. Компілятор дає нам змогу обрати одразу тип створюваного проекту, буде опція «App», в ній користувач може створити додаток із звичайними параметрами (Рис. 3.5).

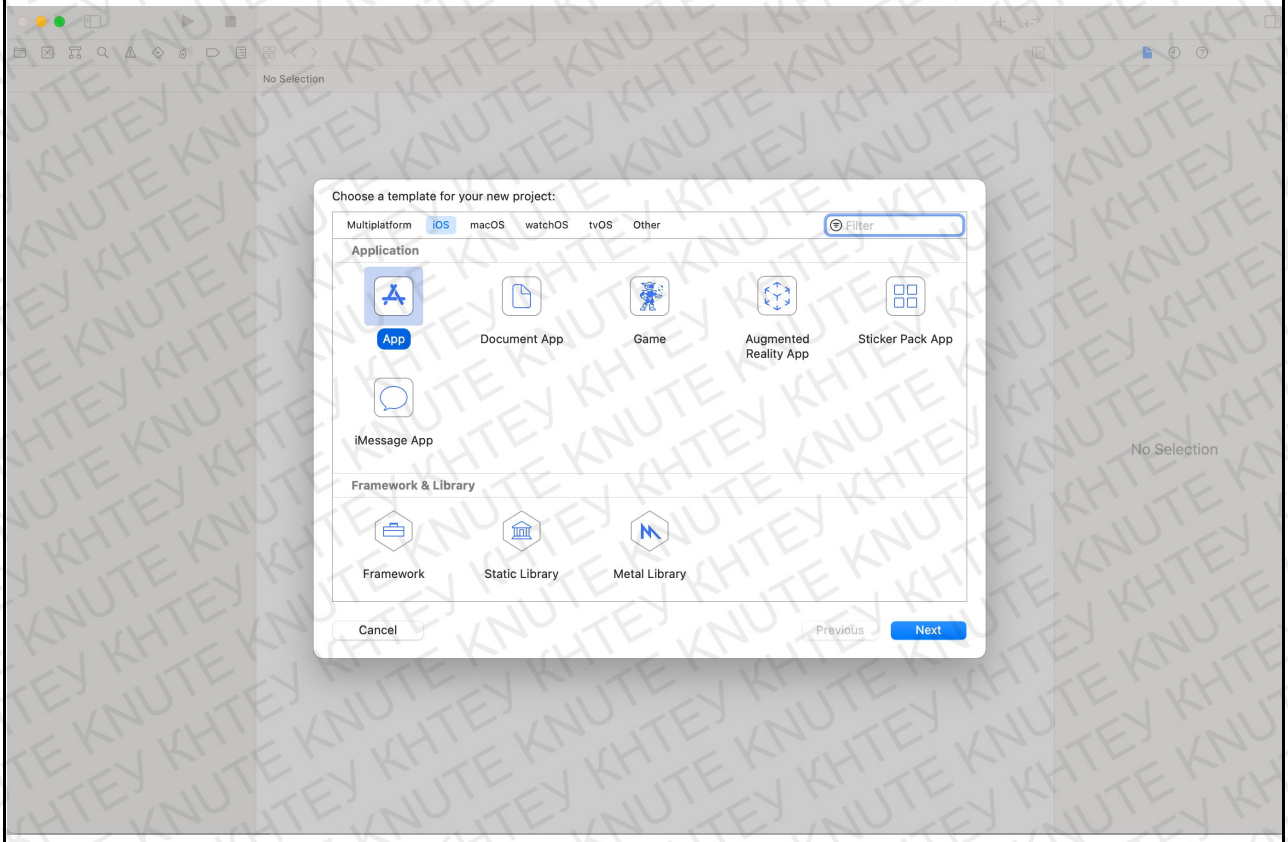

Рис 3.5 створення проекту в Xcode

Далі ми мусимо ввести назву продукту, назву компанії, ідентифікатор компанії, ідентифікатор проекту. Обрати опції інтерфейсу, в залежності від опцій інтерфейсу - життєвий цикл нашого додатку. Також можна ввімкнути опції налаштувати локальну базу данних одразу(Рис. 3.6).

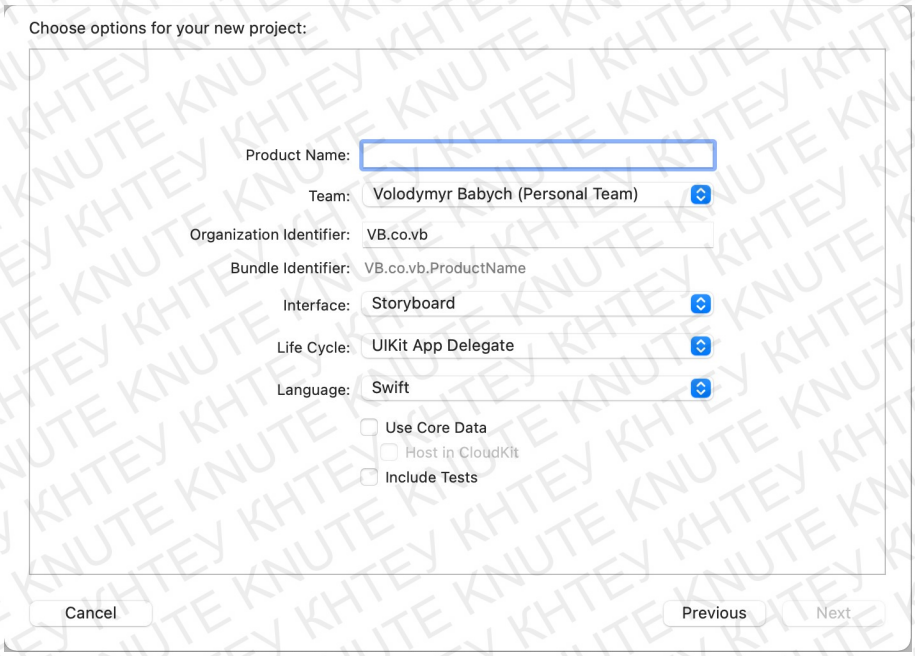

Рис 3.6 створення проекту в Xcode

Вкажемо в якому місці проект бути зберігатись. Одразу можна вибрати опцію яка створить автоматично локальний Git repo проекту(Рис 3.7).

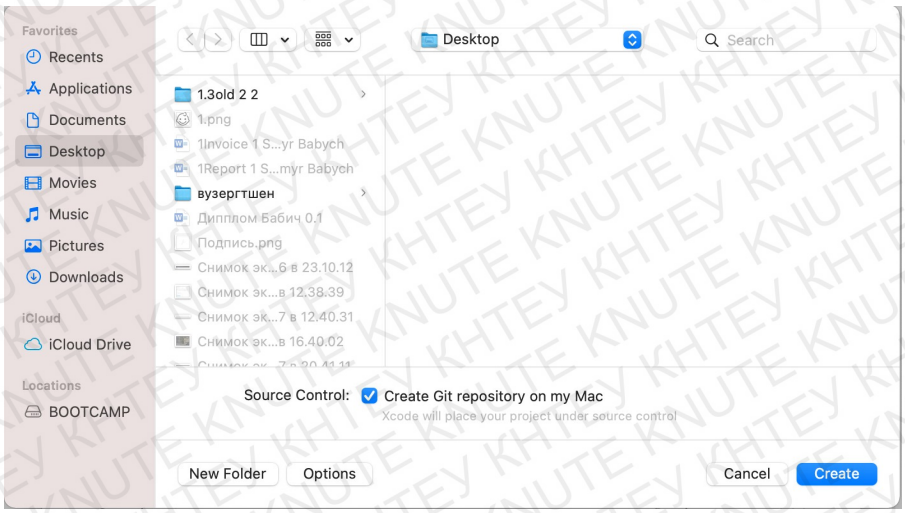

Рис 3.7 створення проекту в Xcode

Для початку нам потрібно створити вікно, в якому батьки зможуть ввести свій унікальний логін та пароль для того щоб авторизуватися. Для цього обираємо файл Main.storyboard(3.8)

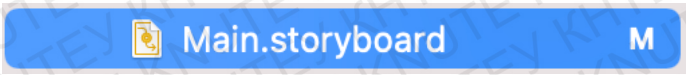

Рис 3.8 файл Main.storyboard

У видкритому вікні Interface Builder ми бачимо екран нашого классу ViewController(3.9)

Рис 3.9 екран классу ViewController

До нього потрібно додати кнопку UIButton, при натисканні на яку користувач авторизуется, та два поля для вводу тексту UITextField які дадуть змогу ввести юзеру свій логін та пароль. Це можна зробити натиснувши кнопку «+» (Рис. Рис. 3.10) у верхньому правому кутку та перетягти потрібний нам елемент до нашого ViewController (Рис. 3.11).

Рис 3.10 кнопка «+»

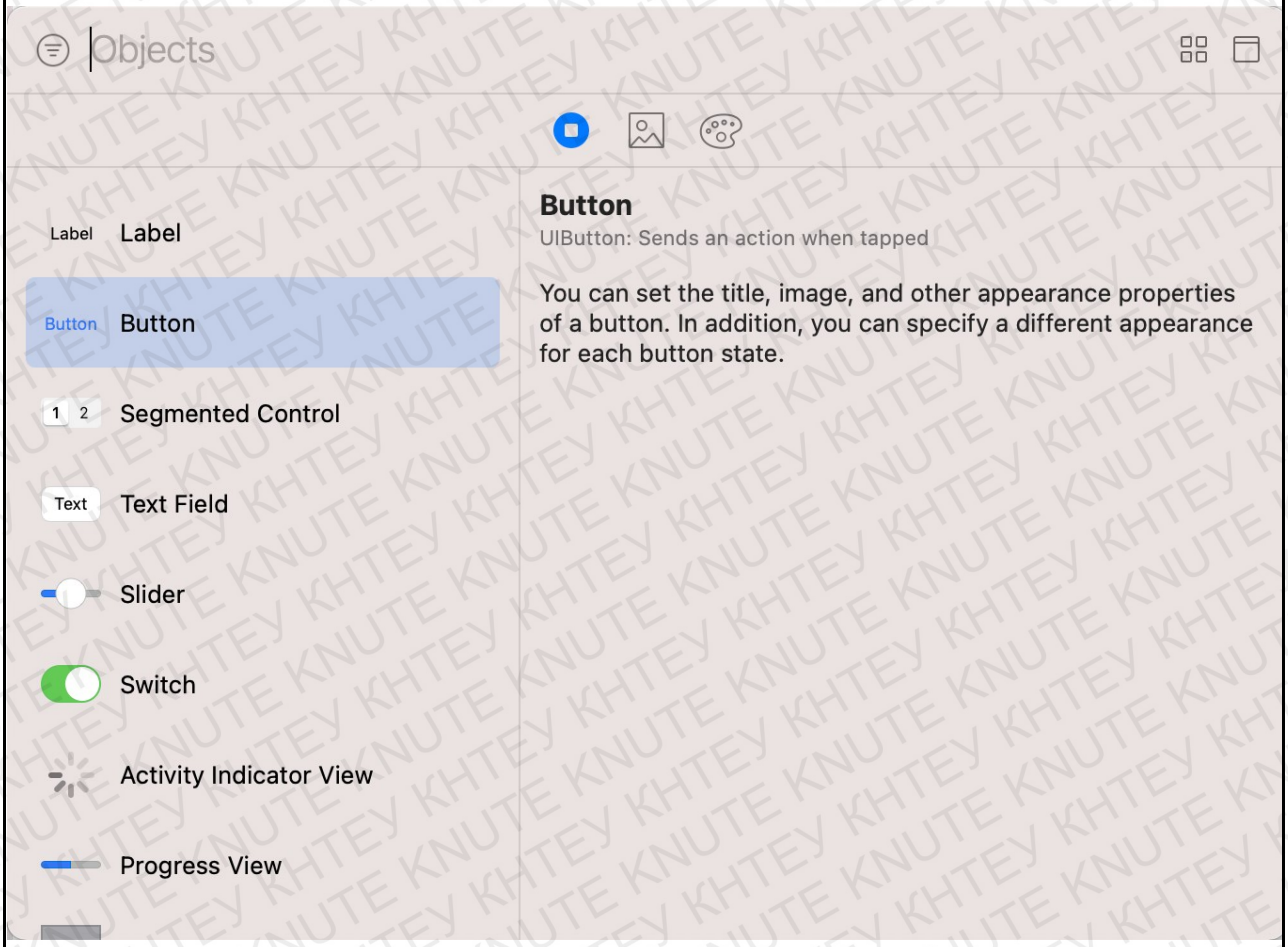

Рис. 3.11 вибір UI елементу

Після додавання елементів наше вікно має такий вигляд (Рис 3.12).

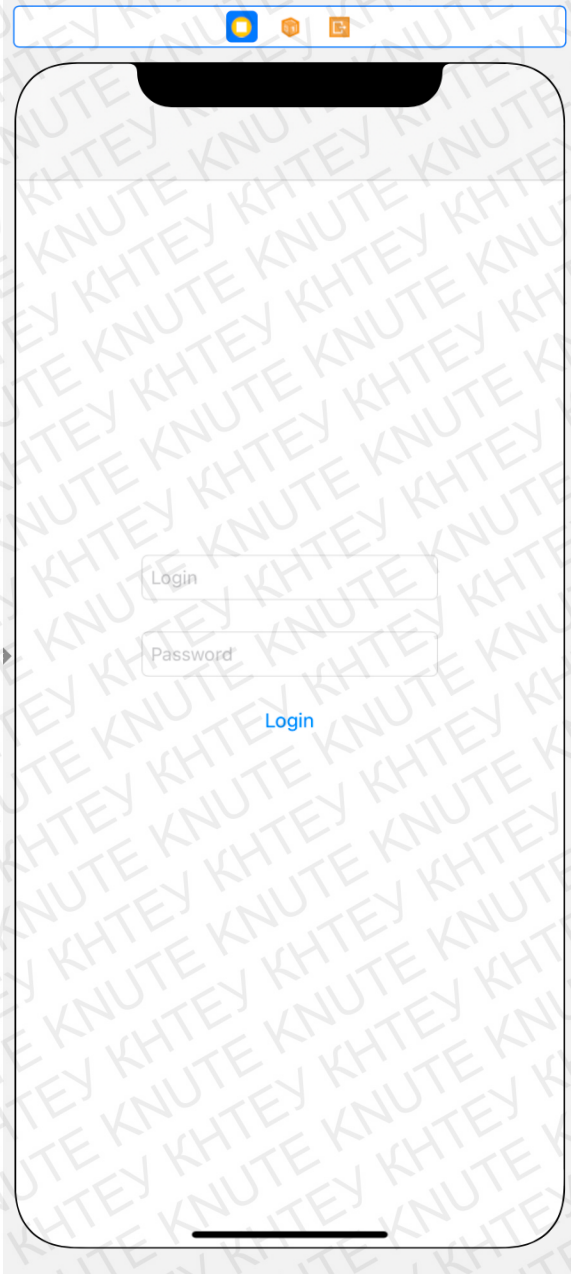

Рис 3.12 єкран ViewController

Натиснувши комбінацію «CMD+R», ми можемо перевірити як наш інтерфейс віглядає на реальному девайсі(Рис 3.13).

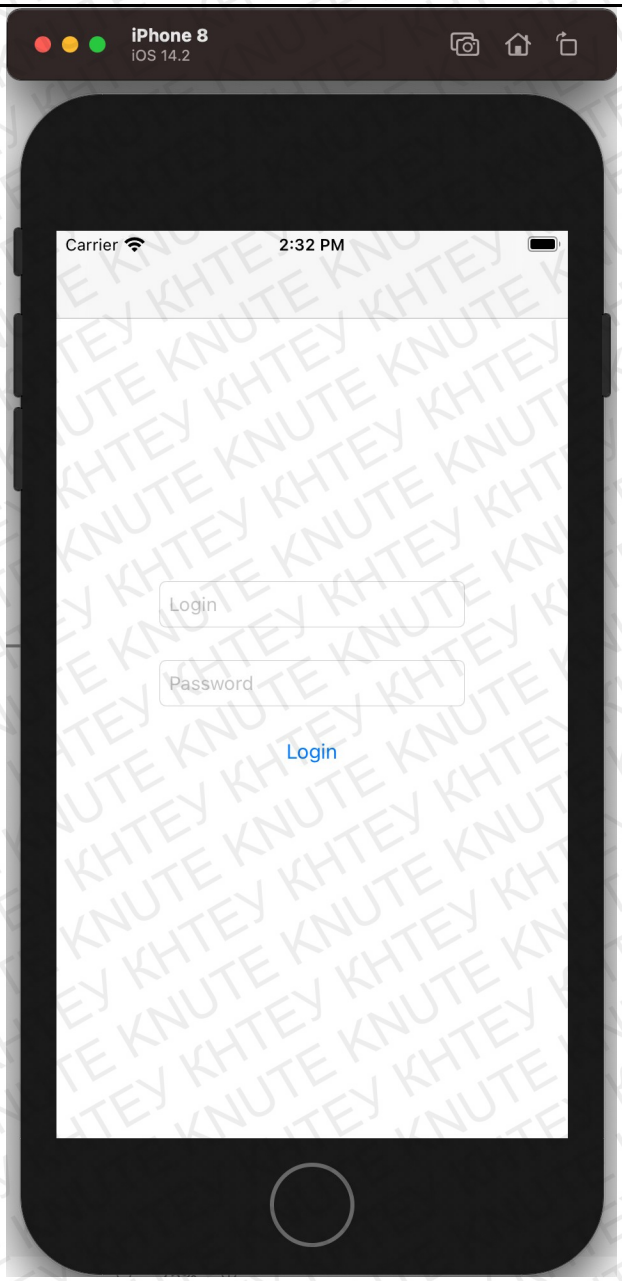

Рис 3.13 єкран ViewController на сімуляторі iPhone 8

На разі всі елементи не мають ніякої логікі і працюють як заглушки. Для того щоб вони працювали так як нам потрібно перейдемо до нашого файлу класса LoginViewController у теці з файлами (Рис 3.14).

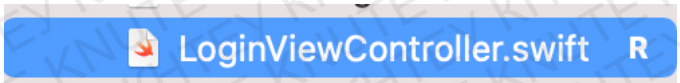

Рис 3.15 файл LoginViewController

У відкритому файлі ми мусимо створити три @IBOutlet для кожного з доданих елементів, та створити звязок с InterfaceBuilder.

```
@IBOutlet weak var loginButton: UIButton!
@IBOutlet weak var passwTextField: UITextField!
@IBOutlet weak var loginTextField: UITextField!
```

Для того щоб користувач міг приховати клавіатуру пристрою, у методі життевого циклу viewDidLoad, який починає працювати під час того як як наш контроллер тільки завантажує UIView, у наших UITextField мусять бути підключені delegate. Делегат - змінна, яка підпорядковується протоколу, який в свою чергу використовується класом, для оповіщення подій або виконання різних підзадач.

```
override func viewDidLoad() {
        super.viewDidLoad()
        passwTextField.delegate = self
        loginTextField.delegate = self
    }
```

`UITextFieldDelegate` має метод який по натиску на кнопку «return» заховає клавіатуру (Рис 3.16).

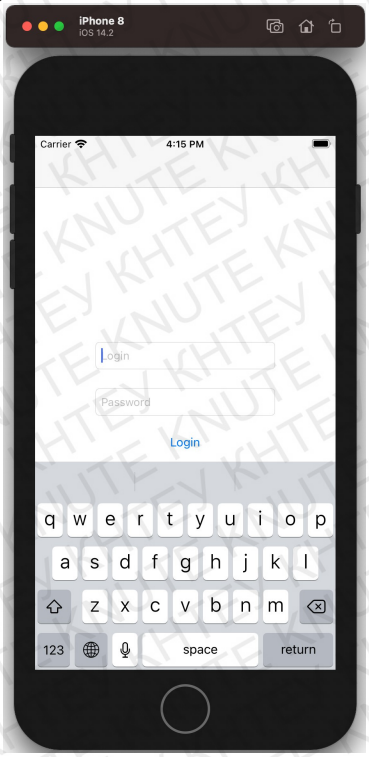

Рис 3.16 клавіатура iOS

Створимо розширення (extension) для нашого контроллера, завдання буде викликати метод textFieldShouldReturn, який буде працювати кожного разу коли користувач натисне на кнопку «return».

```
extension LoginViewController: UITextFieldDelegate{

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()//

        return true
    }
}
```

При натисканні на кнопку Login, користувач мусить відправити на сервер свої данні та при успішній авторизації перейти на наступний рівень. Нам потрібно створити запит до сервера в якому мы відправимо дві строки на опрацювання. Створимо новий класс який буде відповідати за роботу з сервером та обробляти результат. В меню программи натискаемо «File -> New-> File», у наступному вікні оберемо категорію Swift file(Рис 3.17).

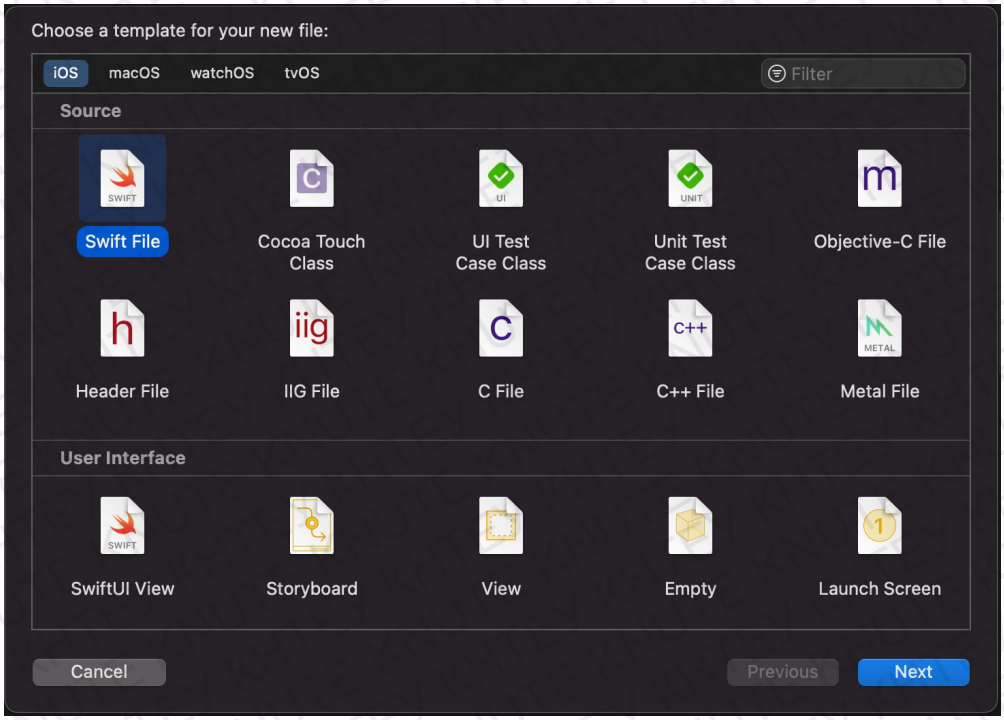

Рис 3.17 створення нового Swift файлу

Створенний файл назвемо ServerManager.swift. Відкриємо його та у порожньому файлі створемо новий клас ServerManager.

```
class ServerManager {
    static let shared = ServerManager()

    func loginRequestWith(login:String, passw:String){

    }
}
```

В середені нашого классу реалізуемо метод loginRequestWith, в який юзер будет передавати свої данні, а метод буде повертати результат.

```
func loginRequestWith(login:String, passw:String, completion:
@escaping (Result<[LoginObjects], NSError>) -> Void){
        guard let url = URL(string: urlLoginString) else { return
}
        let parameters = ["login": login, "passw": passw]
        let session = URLSession.shared
        var request = URLRequest(url: url)
        request.httpMethod = "POST"
        request.setValue("Application/json", forHTTPHeaderField:
"Content-Type")
        request.httpBody = try!
JSONSerialization.data(withJSONObject: parameters, options: [])
        session.dataTask(with: request) { (data, _, error) in
            guard let data = data else {return}

            do {
                let parsedObject = try
JSONDecoder().decode([LoginObjects].self, from: data)

                completion(.success(parsedObject))
            } catch {
                print(error)
                completion(.failure(error as NSError))
            }
        }.resume()
    }
}
```

Для більш зручного парсингу данних ми створили окрему структуру LoginObjects, в якій ми будемо зберігати отримані данні.

```
struct LoginObjects: Decodable {
        let userName:String
        let userImage:String
}
```

Тепер ми повинні викликати цю функцію натиснувши на кнопку «Login», та отримати данні. Для цього створимо @IBAction метод в якому ми буду опрацьовувати кожне натискання на кнопку.

```
@IBAction func loginButtonTapped(_ sender: UIButton) {
    }
```

В методі loginButtonTapped викличемо функцію loginRequestWith.

```
@IBAction func loginButtonTapped(_ sender: UIButton) {
        ServerManager.shared.loginRequestWith(login:
loginTextField.text!, passw: passwTextField.text!) { (result) in
            switch result{
            case .success(let obj):
                print(obj.userName)
            case .failure(_):
                print("error")
            }
        }
    }
```

При успішному виконанні функції створимо код який зможе перенаправити на наступний екран.

```
@IBAction func loginButtonTapped(_ sender: UIButton) {
        ServerManager.shared.loginRequestWith(login:
loginTextField.text!, passw: passwTextField.text!) { (result) in
            switch result{
            case .success(let obj):
                print(obj.userName)
                let childrensVC =
ChildrensViewController(nibName: "ChildrensViewController",
bundle: nil)
                self.navigationController?.viewControllers =
[childrensVC]
            case .failure(_):
                print("error")
            }
        }
    }
```

В цілому наш контроллер мусить мати такий вигляд:

```
import UIKit

class LoginViewController: UIViewController {

    @IBOutlet weak var loginButton: UIButton!
    @IBOutlet weak var passwTextField: UITextField!
    @IBOutlet weak var loginTextField: UITextField!

    override func viewDidLoad() {
        super.viewDidLoad()
        passwTextField.delegate = self
        loginTextField.delegate = self
    }

    @IBAction func loginButtonTapped(_ sender: UIButton) {
        ServerManager.shared.loginRequestWith(login:
loginTextField.text!, passw: passwTextField.text!) { (result) in
            switch result{
            case .success(let obj):
                print(obj.userName)
                let childrensVC =
ChildrensViewController(nibName: "ChildrensViewController",
bundle: nil)
                self.navigationController?.viewControllers =
[childrensVC]
            case .failure(_):
                print("error")
            }
        }
    }
}
extension LoginViewController: UITextFieldDelegate{

    func textFieldShouldReturn(_ textField: UITextField) -> Bool
{
        textField.resignFirstResponder()//

        return true
    }
}
```

Отримавши данні створимо наступне вікно в якому батьки матимуть змогу побачити табличку з дітьми. Для цього ми знову мусимо створити новий файл, але на цей раз ми створимо CocoaTouch Class (Рис 3.18).

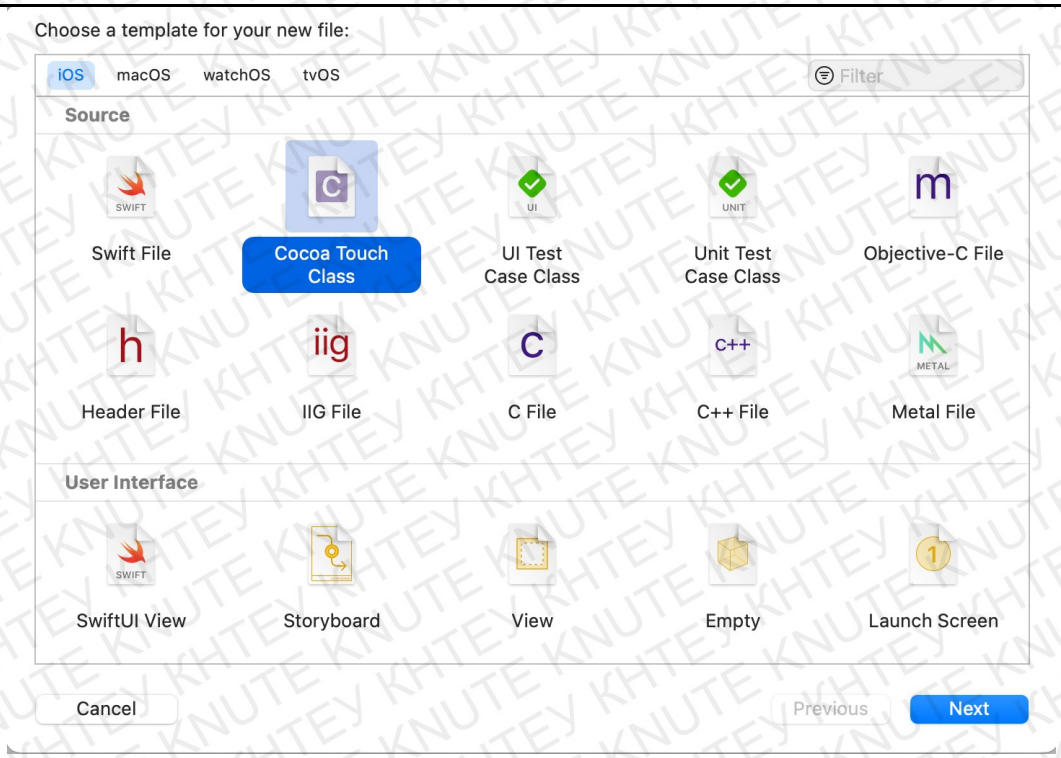

Рис 3.18 створення нового Cocoa Touch файлу

У наступному вікні вкажемо імя класу, спадкоємця і створимо Xib файл в якому ми зможемо редагувати UI елементи(Рис 3.19).

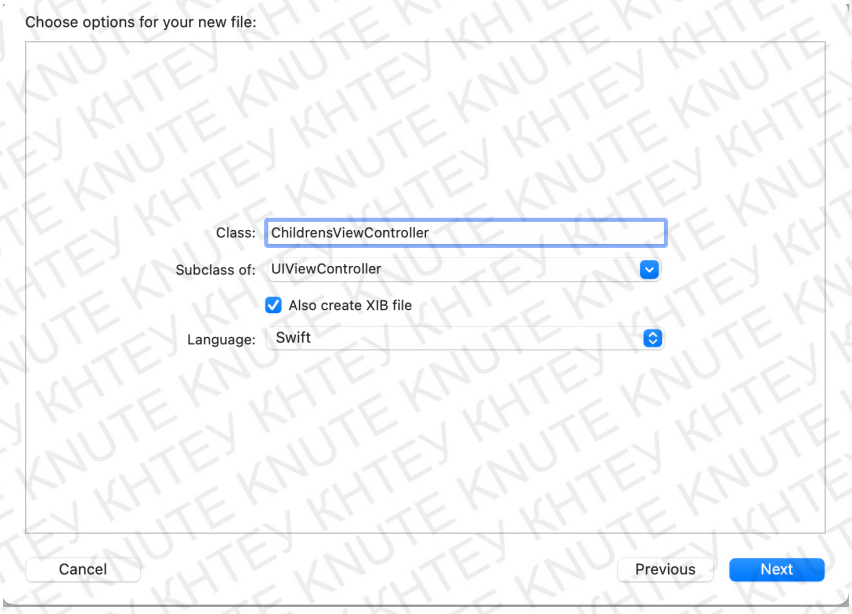

Рис 3.19 створення нового Cocoa Touch файлу

По закінченню операції у стеку файлів у нас зьявлятся два нових файла ChildrensViewController.swift та ChildrensViewController.xib.

Для відображення данних ми будемо використовувати UITableView класс.

```
let childrensTableView = UITableView()
```

Для того щоб додати таблицю на екран використаємо наступний код.

```
let childrensTableView = UITableView()
 override func viewDidLoad() {
        super.viewDidLoad()

        self.childrensTableView.frame = CGRect(x: 0, y: 0, width:
self.view.frame.size.width, height: self.view.frame.size.height)

      self.view.addSubview(childrensTableView)
   }
```

TableView як і TextField також має змінну delegate для обробки данних існує змінна dataSource.

```
self.childrensTableView.dataSource = self
self.childrensTableView.delegate = self
```

Створимо розширення extension.

```
extension ChildrensViewController: UITableViewDataSource{}
extension ChildrensViewController: UITableViewDelegate{}
```

Для того щоб мы могли відобразити данні ми мусимо їх десь взяти. У класі NetworkManager ми отримуемо данні, але ніде їх не сберегли. Для вирішення цієї проблеми нам необхідний менеджер який буде зберігати отриманні данні. Створимо новий Swift file та назвемо його DataManager.

```
class DataManager {
    static let shared = DataManager()
    var data:LoginObjects?
}
```

У змінній data ми будемо зберігати наші данні з сервера. Тому потрібно трохи змінити запит.

```
@IBAction func loginButtonTapped(_ sender: UIButton) {
        ServerManager.shared.loginRequestWith(login:
loginTextField.text!, passw: passwTextField.text!) { (result) in
            switch result{
            case .success(let obj):
                DataManager.shared.data = obj
                let childrensVC =
ChildrensViewController(nibName: "ChildrensViewController",
bundle: nil)
                self.navigationController?.viewControllers =
[childrensVC]
            case .failure(_):
                print("error")
            }
        }
    }
```

Тепер ми можемо написати код, який буде відображати данні в нашій таблиці.

```
extension ChildrensViewController: UITableViewDataSource{

    func tableView(_ tableView: UITableView,
numberOfRowsInSection section: Int) -> Int {
        return DataManager.shared.data.count
    }
    func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {
        let cell = UITableViewCell(style: .default,
reuseIdentifier: "cellId")
        cell.textLabel?.text =
DataManager.shared.data[indexPath.row]
        cell.imageView?.image = UIImage(named: "1.png")
        cell.imageView?.contentMode = .scaleAspectFit
        return cell
    }
}
```

Для кожного row в таблиці ми призначаемо данні з массива по індексу цього row.

Для завершення давайте додамо ще кнопку для виходу з додатку.

```
let logoutButton: UIBarButtonItem = UIBarButtonItem(title:
"Logout", style: .done, target: self, action:
#selector(logoutTapped))

self.navigationItem.rightBarButtonItem = logoutButton
```

Тепер створимо метод logoutTapped, який буде повертати нас на попередній екран.

```
@objc func logoutTapped(){
        let mainStoryboard: UIStoryboard = UIStoryboard(name:
"Main", bundle: nil)
        let loginVC =
mainStoryboard.instantiateViewController(identifier:
"LoginViewController")
        self.navigationController?.viewControllers = [loginVC]
    }
```

Але цього замало сервер також повинен знати що користувач виповнив функцію Logout. Повернемось до нашого ServerManager, та створимо метод logout.

```
func logout( completion: @escaping (Result<LogoutObjects,
NSError>) -> Void){
        guard let url = URL(string: urlLogoutString) else
{ return }
        let session = URLSession.shared
        var request = URLRequest(url: url)
        request.httpMethod = "GET"
        session.dataTask(with: request) { (data, _, error) in
            guard let data = data else {return}

            do {
                let parsedObject = try
JSONDecoder().decode(LogoutObjects.self, from: data)
                completion(.success(parsedObject))
            } catch {
                print(error)
                completion(.failure(error as NSError))
            }
        }.resume()
 }
```

Та нову модель LogoutObjects.

```
struct LogoutObjects: Decodable {
        let success:String
}
```

Додамо новий метод до виконання кнопки Logout.

```
@objc func logoutTapped(){
        ServerManager.shared.logout { (result) in
            switch result{
            case.success(let obj):
                print(obj.success)
                let mainStoryboard: UIStoryboard =
UIStoryboard(name: "Main", bundle: nil)
                let loginVC =
mainStoryboard.instantiateViewController(identifier:
"LoginViewController")
                self.navigationController?.viewControllers =
[loginVC]
            case .failure(let error):
                print(error)
            }
        }
    }
```

Давайте перевіремо як виглядає наш новий екран(Рис 3.20).

Рис 3.20 перегляд нового єкрану

Тепер коли ми бачимо дітей, нам потрібно мати данні про них. У кожної дитини є свій розклад, свое меню. Нам потрібно створити новий запит, в якому ми будемо отримувати доступні опції для конкректної дитини.

В ServerManager створимо новий метод.

```
func getChildOptions(child:String, completion: @escaping
(Result<ChildOptions, NSError>) -> Void){
        guard let url = URL(string: urlOptionsString) else
{ return }
        let parameters = ["child": child]
        let session = URLSession.shared
        var request = URLRequest(url: url)
        request.httpMethod = "POST"
        request.setValue("Application/json", forHTTPHeaderField:
"Content-Type")
        request.httpBody = try!
JSONSerialization.data(withJSONObject: parameters, options: [])
        session.dataTask(with: request) { (data, _, error) in
            guard let data = data else {return}

            do {
                let parsedObject = try
JSONDecoder().decode(ChildOptions.self, from: data)

                completion(.success(parsedObject))
            } catch {
                print(error)
                completion(.failure(error as NSError))
            }
        }.resume()
    }
```

Створимо нову модель.

```
struct ChildOptions: Decodable {
    let videoUrl:String
    let weekMenuImage:String
    let schedule:String
}
```

Тепер нам потрібно по натиску на row нашої TableView, відсилати запит на сервер і якщо запит позитивний переходити на наступне вікно з опціями.

В розширенні UITableViewDelegate застосуємо метод didSelectRowAt, який спрацьовує кожен раз коли ми натискаємо на row, та повертає нам indexPath.

```
extension ChildrensViewController: UITableViewDelegate{
    func tableView(_ tableView: UITableView, didSelectRowAt
indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)

    }
}
```

В цей метод ми додамо код запиту та код переходу на настурний екран.

```
extension ChildrensViewController: UITableViewDelegate{
    func tableView(_ tableView: UITableView, didSelectRowAt
indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
        pushChildrenVc(index: indexPath.row)
    }
}

func pushChildrenVc(index:Int){
        ServerManager.shared.getChildOptions(child:
DataManager.shared.data[index]) { (result) in
            switch result{
            case .success(let options):
                DataManager.shared.options = options
                let childrenVC = ChildrenViewController(nibName:
"ChildrenViewController", bundle: nil)

self.navigationController?.pushViewController(childrenVC,
animated: true)
            case .failure(let error):
                print(error)
            }
        }
    }
```

Для збереження данних з опціями, створемо нову змінну у DataManager.

```
class DataManager {
    static let shared = DataManager()
    var data:LoginObjects?
    var options: ChildOptions!
}
```

Тепер створимо новий клас ChildrenViewController(Рис 3.21).

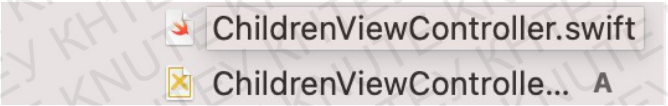

Рис 3.21 створення класу ChildrenViewController

Цей екран мусить відображати опції які ми отримали від сервера. Данні різного типу, тому створимо на ньому також TableView і кожнен рядок таблиці матиме назву опції.

```
let optionsTableView = UITableView()
let optionsTitlesArray = ["Watch video", "Show Menu", "Show
Shedule"]

override func viewDidLoad() {
        super.viewDidLoad()

        self.optionsTableView.frame = CGRect(x: 0, y: 0, width:
self.view.frame.size.width, height: self.view.frame.size.height)

        self.optionsTableView.dataSource = self
        self.optionsTableView.delegate = self
        self.view.addSubview(optionsTableView)
    }

extension ChildrenViewController: UITableViewDataSource{
    func tableView(_ tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        return optionsTitlesArray.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {
        let cell = UITableViewCell(style: .default,
reuseIdentifier: "cellId")
        cell.textLabel?.text = optionsTitlesArray[indexPath.row]
        return cell
    }
}
```

Тепер наш єкран має такий вигляд(Рис 3.22).

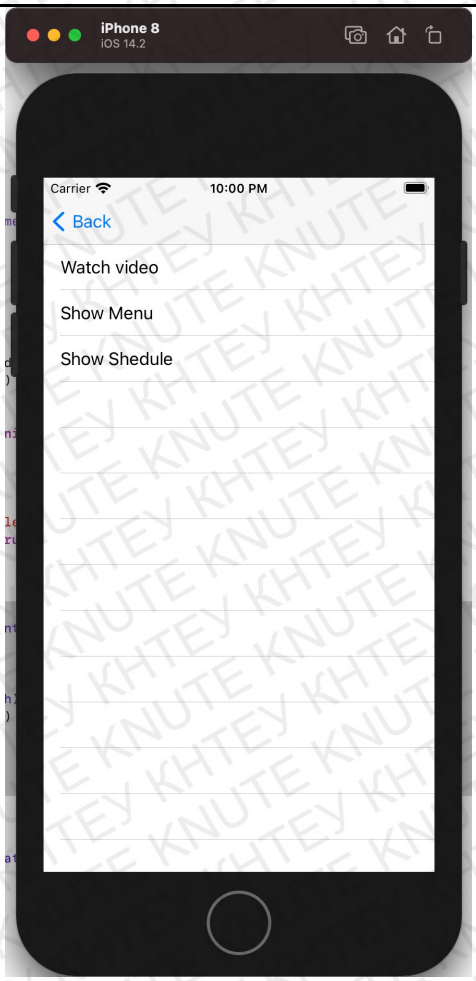

Рис 3.22 екран ChildrenViewController

Тепер по натисканнню на кожний рядок, маемо створити перехід на потрібний нам екран. Напишемо перевірку яка будет перевіряти індекс натиснутого рядка та перенаправляти на потрібний екран.

```
extension ChildrenViewController: UITableViewDelegate{
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath:
IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
        switch indexPath.row {
        case 0:
            pushVideoVc()
        case 1:
            pushMenuVc()
        case 2:
            pushScheduleVc()
        default:
            print("defaultValue")
        }
    }
}
```

```
func pushVideoVc(){
        let videoVC = VideoViewController(nibName:
"VideoViewController", bundle: nil)
        self.navigationController?.pushViewController(videoVC,
animated: true)
    }
    func pushMenuVc(){
        let menu = MenuViewController(nibName:
"MenuViewController", bundle: nil)
        self.navigationController?.pushViewController(menu,
animated: true)
    }

    func pushScheduleVc(){
        let scheduleVC = ScheduleViewController(nibName:
"ScheduleViewController", bundle: nil)
        self.navigationController?.pushViewController(scheduleVC,
animated: true)
    }
```

Створимо новий VideoViewController (Рис 3.23)

Рис 3.23 створення VideoViewController

Завдання цього контроллера відображати список відеозаписів які приходят з сервера, та програвати їх. Створемо TableView.

```
let tableView = UITableView()
    var videoTitlesArray = [String]()
    override func viewDidLoad() {
        super.viewDidLoad()
        tableView.frame = self.view.frame
        self.view.addSubview(tableView)

        self.tableView.dataSource = self
        self.tableView.delegate = self

    }
```

```
extension VideoViewController: UITableViewDataSource{
    func tableView(_ tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        return videoTitlesArray.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {
        let cell = UITableViewCell(style: .default,
reuseIdentifier: "cellId")
        cell.textLabel?.text = videoTitlesArray[indexPath.row]
        return cell
    }
}
```

Тепер на екрані ми бачимо таблицю з тайтлами відеозаписів (Рис 3.24).

Рис 3.24 VideoViewController

Тепер наше завдання програти відео натиснувши на рядок таблиці. Для цього там потрібно створити AVPlayer який буде програвати видео по заданому URL.

```
private func playVideo(index:Int) {
            let urlString = videoUrlsArray[index]
            let url = URL(string: urlString)
            let player = AVPlayer(url: url!)
            let playerController = AVPlayerViewController()
            playerController.player = player
            present(playerController, animated: true) {
                player.play()
```

Додамо цю функцію до методу didSelectRowAt.

```
extension VideoViewController: UITableViewDelegate{
    func tableView(_ tableView: UITableView, didSelectRowAt
indexPath: IndexPath) {
        tableView.deselectRow(at: indexPath, animated: true)
        playVideo(index: indexPath.row)
    }
}
```

Тепер нам потрібно створити екрани в яких будут відображатись Image з розкладом занят, там меню страв. Принцип їх роботи абсолютно однаковий, але в подальшому кожен з цих екранів буде розширюватись тому я створив два різних. Розглянемо роботу одного с зних. Створимо MenuViewController для відображення меню страв(Рис 3.25).

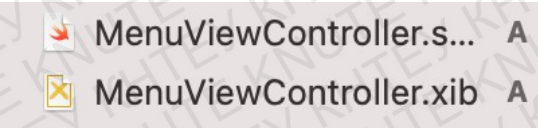

Рис 3.25 створення MenuViewController

Нам необхідно створити UIImageView в якому ми будемо відображати image.

```
override func viewDidLoad() {
        super.viewDidLoad()
        let imageView = UIImageView(frame: self.view.frame)
        self.view.addSubview(imageView)
        imageView.image = UIImage(named: weekMenuImage)
    }
```

Наш екран тепер має такий вигляд (Рис 3.26)

Рис 3.26 MenuViewController

3.4 Висновки до розділу 3

Отже ми дізнались що таке Xcode. Xcode – це пакет інструментів для розробки додатків під Mac OS X iOS, розроблений Apple. Розглянули мови для розробки, та обрали Swift через низку переваг. Обрано архітектуру MVC, створили серверну, та кліентську частину мобільного додатку.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

З проведених досліджень можна зробити наступні висновки:

1. Проведено аналіз розробки мобільного додатку, та вибрано клієнт-серверну архітектуру застосунку, яка є оптимальною для реалізації поставлених завдань.

2. Проводено аналіз у пошуках найпоширеніших оперційних систем.

3. Проведено аналіз закладів дошкільної освіти в Україні та виявлено їх недоліки.

4. В якості операційної системи обрано ОС iOS(мобільна система) як одна з найпопулярніших операційних систем.

5. Проведений аналіз засобів розробки показав, що найзручнішим для реалізації поставлених завдань є Xcode.

6. В процесі виконання магістерської роботи розроблено систему в якій батьки матимуть змогу контролювати своїх дітей, та втручатись у процес виховання.

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="4" rowspan="3">КНТЕУ 121 06з-1.МР</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Зм.</td><td>Аркуш</td><td>№ докум.</td><td>Підпис</td><td>Дата</td></tr>
<tr><td>Зав. каф.</td><td></td><td>Криворучко О.В.</td><td></td><td>30.10.20</td><td rowspan="3">«Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»</td><td>Стадія</td><td>Аркуш</td><td>Аркушів</td></tr>
<tr><td>Керівник</td><td></td><td>Цензура М.А.</td><td></td><td>30.10.20</td><td>ВП</td><td>75</td><td>78</td></tr>
<tr><td>Гарант</td><td></td><td>Криворучко О.В.</td><td></td><td>30.10.20</td><td rowspan="2">Факультет інформаційних технологій 2м курс, 6з група</td></tr>
<tr><td>Розробив</td><td></td><td>Бабич В.О.</td><td></td><td>30.10.20</td><td>Висновки та пропозиції</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Вікіпедія // https://uk.wikipedia.org/wiki/Мобільний_застосунок;
2. Fidalgo, Antonio. Pushed News: When the News Comes to the Cellphone. / Brazilian Journalism Research. – 2009 – No 2/5. – P. 113–124.
3. Wolf, Cornelia. Revolution in Journalism? Mobile Devices As a New Means of Publishing. / C. Wolf, R. Hohlfeld // In Images in Mobile Communication, edited by Corinne Martin and Thilo von Pape: VS Verlag für Sozialwissenschaften. / C. Wolf,

R. Hohlfeld. – Germany, 2012. – p. 32–35

4. Liu, Cheng. Mobile News in Chinese Newspaper Groups: A Case Study of Yunnan Daily Press Group. / C. Liu, A. Bruns. // In Mobile Technologies: From Telecommunications to Media / C. Liu, A. Bruns. – New York: Routledge, 2009. – p. 187–201.
5. Westlund, Oscar. Digital Journalism. / O. Westlund // Department of Journalism, Media and Communication, University of Gothenburg 2013 / O. Westlund. – Gothenburg, 2013. – p. 5–10.
6. Shelley, Seale. Emerging Mobile Strategies for News Publishers. / S. Seale. // International Newsmedia Marketing Association (INMA) / INMA – New York – 2012. –No 1/1. – P. 39–47.
7. Göteborgs universitet [Electronic resource] / Westlund, Oscar. – Cross-media News Work: Sensemaking of the Mobile Media (R)evolution. JMG Book Series – Modeofaccess:https://gupea.ub.gu.se/bitstream/2077/28118/1/gupea_2077_28118_1.pdf.

8. Westlund, Oscar. Användning av mobilen för information och kommunikation[Using the Mobile for Information and Communication]. / O. Westlund // In I mediernas skugga [Shadowed by the Media] / L.Weibull, A. Bergström, and H. Oscarsson. – Gothenburg, 2012. – p. 6–15
9. Westlund, Oscar. New(s) Functions for the Mobile. / O. Westlund // New Media & Society / University of Gothenburg. – Gothenburg, 2010. – p. 91– 108.
10. Westlund, Oscar. Diffusion of Internet for Mobile Devices in Sweden. // Nordic and Baltic Journal of Information and Communications Technologies – 2008. – No 1/2. – P. 39–47.
11. https://www.netmarketshare.com/operating-system-market-share.aspx?options=
12. Як працює Android, частина 1 [Электронный ресурс] // Habrahabr.https://habr.com/company/solarsecurity/blog/334796/
13. Як працює Android, частина 3 [Электронный ресурс] // Habrahabr. https://habr.com/company/solarsecurity/blog/338494/
14. https://uk.wikipedia.org/wiki/Objective-C
15. https://uk.wikipedia.org/wiki/Swift_(мова_програмування)
16. https://habr.com/company/badoo/blog/281162/
17. https://habr.com/post/324414/
18. http://ir.nmu.org.ua/bitstream/handle/123456789/151453/Козир_AB.pdf?sequence=1&isAllowed=y
19. https://developer.apple.com/xcode/interface-builder/
20. https://stfalcon.com/en/blog/post/PHP-spheres-of-use
21. https://mon.gov.ua/ua/news/z-pochatku-2020-roku-v-dityachih-sadkah-stvoreno-ponad-10-tisyach-novih-misc-operativni-dani-mon
22. https://uk.wikipedia.org/wiki/Дошкільний_навчальний_заклад

23. https://nv.ua/ukr/opinion/dityachi-sadki-v-ukrajini-z-yakimi-problemami-zishtovhuyutsya-vihovateli-ta-batki-sergiy-babak-ostanni-novini-50072912.html
24. https://khreschatyk.news/pryvatni-dytyachi-sadochky-kyyeva-poryatunok-dlya-batkiv-chy-skrynka-pandory/
25. https://index.minfin.com.ua/labour/salary/average/
26. https://khreschatyk.news/pryvatni-dytyachi-sadochky-kyyeva-poryatunok-dlya-batkiv-chy-skrynka-pandory/
27. https://tsn.ua/ukrayina/smert-odnorichnoyi-divchinki-u-sadku-zaporizhzhya-vihovatelka-zatulila-yiy-rot-podushkoyu-ta-znischuvala-dokazi-1560438.html
28. https://www.babysdays.com/product-information/what-is-babys-days

ТЕХНІЧНЕ ЗАВДАННЯ

Однією з найголовніших запорук відповідності вимогам та високої якості розробленого програмного рішення є грамотно спроектоване до початку розробки технічне завдання. Отже, далі розкриємо невід'ємні положення технічного завдання до розробки iOS-додатку для закладу дошкільної освіти.

1. Загальні відомості

Назва проекту: Child Saver.

Планові строки розробки: серпень 2019 – вересень 2019.

Потенційні користувачі: Батьки та заклади дошкільної освіти.

Призначення програмного рішення: надати можливість батькам слідкувати за розвитком дітей у садочках.

Мета створення програмного рішення: зменшити кількість форсмажорів у закладах дошкільної освіти.

2. Структура проекту

Серверна частина, яка має методи авторизації та обробки данних користувача. Кліентська частина яка працює з сервером оброблює данні та їх відображає на приладі.

3. Функціональні вимоги до додатку

Користувач мусить мати змогу авторизоватися, та взаємодіяти з доступними для нього опціями.

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td colspan="4" rowspan="2">КНТЕУ 121 06з-1.МР</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>Зм.</td><td>Аркуш</td><td>№ докум.</td><td>Підпис</td><td>Дата</td></tr>
<tr><td colspan="2">Зав. каф.</td><td>Криворучко О.В.</td><td></td><td>19.10.20</td><td colspan="2" rowspan="3">«Розробка мобільного додатку на платформі iOS для закладу дошкільної освіти «Антошка»»</td><td>Стадія</td><td>Аркуш</td><td>Аркушів</td></tr>
<tr><td colspan="2">Керівник</td><td>Цензура М.А.</td><td></td><td>19.10.20</td><td>ТЗ</td><td>79</td><td>78</td></tr>
<tr><td colspan="2">Гарант</td><td>Криворучко О.В.</td><td></td><td>19.10.20</td><td colspan="3" rowspan="2">Факультет інформаційних технологій 2м курс, 6з група</td></tr>
<tr><td colspan="2">Розробив</td><td>Бабич В.О.</td><td></td><td>19.10.20</td><td colspan="2">Огляд та засоби розробки мобільного додатку</td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td><td colspan="5"></td></tr>
</table>