Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____
(повна назва)
Кафедра ____Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ___другий (магістерський)_____
(рівень вищої освіти)

Модель аналізу виконання графіку навчального процесу з використанням мобільного застосунку
(тема)

Виконав: студент 2 курсу, групи СКСм-21-1
Старіна В.Д.
(прізвище, ініціали)
Спеціальність _____
123 Комп'ютерна інженерія
(код і повна назва спеціальності)
Тип програми _____
освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма_____
Спеціалізовані комп'ютерні системи
(повна назва освітньої програми)
Керівник роботи _____доц. Шкіль О.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Чумаченко С.В._____
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проектування обчислювальної техніки_____

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_ Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри __  _____
(підпис)

«_02__» __вересня__ 2022 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту__Старині Владиславу Дмитровичу_____
(прізвище, ім'я, по батькові)

Тема роботи (проекту)_ Модель аналізу виконання графіку навчального процесу з використанням мобільного застосунку_____
Analysis Model for the Schedule Execution of the Educational Process Using Mobile Application_____

затверджена наказом по університету від «__14_»__11__2022 р. № _1478 Ст_____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи (проекту)_____
Графік навчання_____
Розклад навчання_____
Середовище XCode_____
Мова програмування Swift_____
_____

4. Перелік питань, що потрібно опрацювати у роботі _____
Моделі процесу навчання_____
Розрахунок прогресу навчання_____
Проектування архітектури мобільного застосунку_____
Процес введення даних_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 20 слайдів

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 01.09.2022

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.09.2022 - 05.09.2022	виконано
2	Пошук та аналіз літературних джерел за темою дипломної роботи	05.09.2022 - 30.09.2022	виконано
3	Визначення основних етапів алгоритмів для оптимізації	01.10.2022 - 15.10.2022	виконано
4	Пошук методів вирішення проблеми	15.10.2022 - 05.11.2022	виконано
5	Програмна реалізація	05.11.2022 - 20.11.2022	виконано
6	Опис програмної реалізації	20.11.2022 - 01.12.2022	виконано
7	Оформлення роботи	01.12.2022 - 15.12.2022	виконано
8	Представлення роботи до захисту	15.12.2022 - 25.12.2022	виконано

Студент _____
(підпис)

Керівник роботи (проекту) _____ доц. кафедри АПОТ Шкіль О.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 61 с., 34 рис., 7 джерел.

SWIFT, XCODE, IOS, SWIFT, UIVIEWCONTROLLER, МОБІЛЬНИЙ ЗАСТОСУНОК, ОБЛІК НАВЧАННЯ, ДЕДЛАЙН, ШКАЛА ПРОГРЕСУ, ГРАФІК НАВЧАННЯ, ДІДЖИТАЛІЗАЦІЯ ОСВІТИ, ПРОГРМНЕ ЗАБЕЗПЕЧЕНЯ

Метою кваліфікаційної роботи є проектування мобільного застосунку в рамках реалізації хмарних сервісів кіберуніверситету для обліку успішності виконання навчального графіку студента з відображенням шкали прогресу.

Облік успішності навчання передбачає покращення процесу самоконтролю студентів та полегшити організацію навчання студентів в період сесії та екзаменів.

Розроблено та протестовано мобільний застосунок, що допомагає слідкувати за вчасним виконанням завдань студентами, а також за їх прогресом у навчанні.

Мобільний застосунок реалізований на базі операційної системи iOS. Програмна реалізація алгоритму додатку здійснена на мові програмування Swift у середовищі розробки XCode.

ABSTRACT

The explanatory note contains: 61 pages, 34 figure, 7 sources according to the list of links.

SWIFT, XCODE, IOS, MOBILE APPLICATION, STUDYING ACCOUNTING, DEADLINE, PROGRESS SCALE, STUDYING SCHEDULE, DIGITALIZATION OF EDUCATION, SOFTWARE

The purpose of the qualification work is to design a mobile application in the implementation of cloud services of the cyber university to account for the success of the student's schedule with the reflection of the scale of progress.

Accounting for academic success involves improving the process of student self-control and facilitate the organization of student learning during sessions and exams.

A mobile application has been developed and tested, which helps to monitor the temporary recognition of students' tasks, as well as their progress in studies.

The mobile application is implemented on the basis of the iOS operating system. The software implementation of the application algorithm is carried out in the Swift programming language in the XCode development environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОС – операційна система;

ПЗ – програмне забезпечення;

API – описання методів за допомогою яких одна програма може комунікувати з іншою (анг., Application programming interface);

CPS – Cyber Physical System;

iOS – це власницька мобільна операційна система від Apple;

IT – інформаційні технології (анг., Information Technologies);

LMS – системам управління навчанням;

SCU – Smart Cyber University .

ВСТУП

Актуальність роботи визначається розвитком кіберфізичного цифрового простору та задачею, пов’язаною з розробкою хмарного кібер-сервісу оцінки якості освітніх процесів та компонентів, онлайн-перевірки завдань, а також підвищенням якості навчального процесу. Практичний аспект також полягає в оптимізації самоорганізації студентів. Через велику кількість навчальних курсів та практичних завдань зростає складність контролю виконання робіт та оцінки часу, який необхідно використати до дедлайнів.

Для вирішення даних проблем можна скористатися нагадуваннями в мобільному телефоні. Це перше та логічне рішення, яке може бути. Телефон майже постійно з людиною, користувач сам може встановлювати події чи дедлайни та видаляти їх, проте в такого рішення є певні недоліки. По-перше, кількість нагадувань буде пропорційна кількості курсів та робіт. Для кожного окремого навчального предмета необхідно встановлювати нове нагадування. Це не є зручним та ефективним. По-друге, окрім нагадувань з навчання в користувача можуть бути інші справи, які необхідно пам’ятати. Тобто, кількість нагадувань ще більша, ніж вже було зазначено, і при цьому всі нагадування перемішані. По-третє, не вирішено питання оцінки того, наскільки успішно виконується курс чи завдання. Немає можливості отримати наочно ступінь прогресу у навчанні. Таким чином, лише нагадуваннями в телефоні неможливо вирішити всі описані проблеми.

В даній кваліфікаційній роботі пропонується створити облік успішності виконання навчального графіку студента у мобільному додатку. Користувач буде мати змогу самостійно встановлювати дедлайни до робіт, зберігати інформацію про вже виконанні завдання та оцінки за них, отримувати заздалегідь повідомлення про термін здачі роботи та наглядну шкалу прогресу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Розумний кіберуніверситет

Мета розумного кіберуніверситету – створення ефективної університетської кіберсистеми управління, що включає інфраструктуру, кадри, відносини та управління, що забезпечує високий рівень наукових та освітніх процесів у відповідно до чинного законодавства для досягнення європейських вимог якості випускників та рівня життя співробітників.

Сутність проекту – приведення всіх компонентів системи (інфраструктура, кадри, управління, відносини та напрямок руху) до чинному законодавству, а також оптимізація структурних підрозділів та науково-освітніх процесів для досягнення європейської якості випускників та рівня життя співробітників.

Завдання актуальні для реконструкції відносин в університетах.

1. Створення метрики для моніторингу та вимірювання ринкової якості суб'єкта: студента, працівника, вченого, структурного підрозділу, університету, міста, країни.

2. Розробка критеріїв оцінювання якості конкретного суб'єкта шляхом метричного порівняння фактичних оцінок з еталонними чи найкращими значення параметрів.

3. Вироблення керуючих регуляторних впливів, орієнтованих на формування roadmap з метою досягнення бажаного рівня якості суб'єкта на основі планування дій сьогоднішнього дня функціональної залежності від замовленого майбутнього.

4. Використання технологій big data та паралельних віртуальних спеціалізованих мультипроцесорів для створення метрично ранжованих розумних інформаційних структур даних та кіберсистем квазіоптимального управління неприродними процесами

5. Написання нового статуту університету відповідно до чинного законодавством.

6. Формування системи керуючих корпоративних положень, регулюючих відносин в організації науково-освітньої діяльності на основі законів та Статуту.

7. Оптимізація структурних підрозділів університету відповідно до принципами доцільності та розумної достатності.

8. Формування компетентнісних метрик для науково-освітніх процесів, співробітників та підрозділів з метою вироблення системи корпоративних положень, що регламентують цифровий моніторинг та кіберуправління ресурсами в університеті

9. Впровадження проекту у життєдіяльність університету шляхом послідовної імплементації напрацьованих відносин та положень у кіберсистему управління.

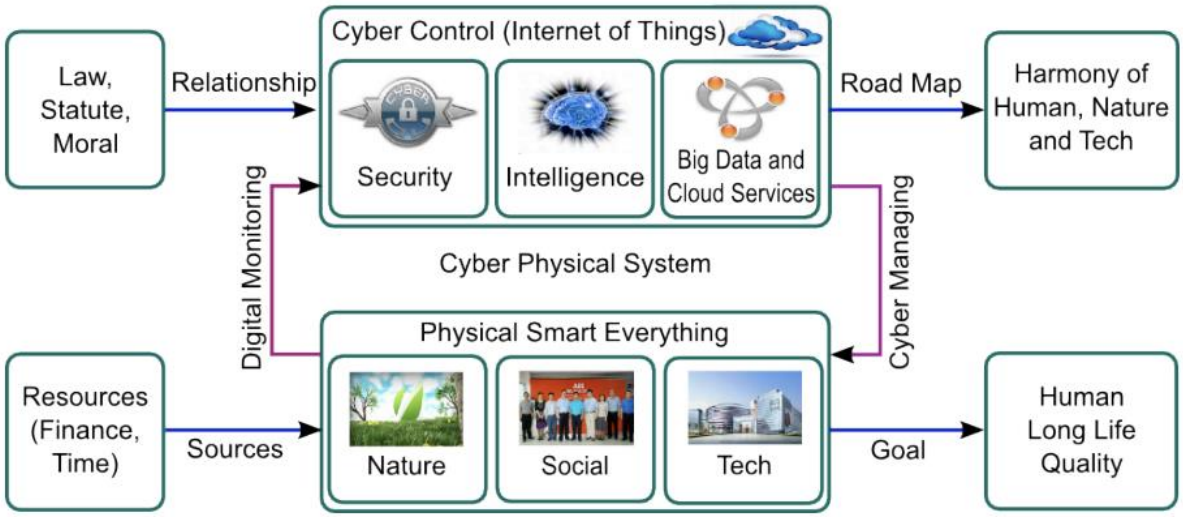

Рисунок 1.1 – Кіберфізична система управління неприродними процесами

Мета, як вихід Cyber Physical System (CPS), функціонально залежить від точного цифрового моніторингу та оптимального кіберуправління віртуальними та фізичними ресурсами, що включають час, гроші, кадри та

матеріали. Головною відмінністю запропонованої масштабованої кіберфізичної системи є відсутність людського фактора в блоці управління (Cyber), що робить її, за конструктивної та гуманної законотворчості, справедливою, ефективною, оптимальною, надійною та захищеною від суб'єктивних помилок менеджера.

Університет як кіберсистема (рис. 1.2) включає: 1) кваліфіковані кадри, 2) розумну інфраструктуру, 3) кібер-управління та -моніторинг, 4) морально-етичні відносини (закони, статут, накази, діловий етикет), 5) напрямок руху – Roadmap (Smart Cyber University) з виділеними зовнішніми ресурсами (абітурієнти, час та гроші) для досягнення мети – забезпечення високої якості життя співробітників та підготовка валідних для ринку спеціалістів. Зробити сприятливий клімат в університеті для фінансових інвестицій, припливу професійних кадрів, рекрутингу найкращих абітурієнтів з усього світу можливе лише за рахунок переформатування всіх основних компонентів ВНЗ як системи, проведення актуальних наукових досліджень світового рівня та впровадження передових освітніх технологій (massively online open courses). Згідно з визначенням ВНЗ-системи оптимізація науково-освітнього процесу та структури підрозділів університету повинна бути спрямована на забезпечення гідного рівня життя членів колективу та високої якості випускників за рахунок: 1) Прозорого морального та матеріального кіберстимулювання на основі кібермоніторингу результативної діяльності вчених; 2) Повсюдного застосування електронного документообігу у технологічні процеси управління університетом; 3) Зменшення чисельності непродуктивного апарату моніторингу та управління шляхом трансформування відповідних кадрів у співробітників кафедр; 4) Зменшення часу на допоміжні процеси, безпосередньо не пов'язані з науково-освітньою діяльністю вчених та співробітників за рахунок впровадження кіберсистеми управління ресурсами (час, гроші, кадри); 5) Визначення збалансованої структури основних виробничих підрозділів на основі врахування ринкових тенденцій попиту на фахівців для створення однакових за значимістю

факультетів та кафедр, що випускають, формують рівні показники результатів науково-освітньої діяльності; 6) Створення доброзичливого клімату морально-етичних відносин для конструктивної творчості вчених та професорів. Сильні факультети та кафедри, розумна інфраструктура та кіберуправління, моральноетичні відносини – сучасний та розумний кіберуніверситет. Реалізувати CPS "розумний кіберуніверситет" для кожного співробітника означає – убезпечити себе від авторитарного свавілля керівників першого рівня, які не завжди законно розподіляють час, гроші та посади. Ректор та проректори, позбавлені чиновницьких привілеїв ручного розподілу ресурсів за середню зарплату вченого, перетворяться на нормальних людей із представницькими функціями підписання документів та наказів, що прозоро формуються кіберсистемою управління. Найважливішим інструментом для формування нових справедливих морально-етичних відносин є експертна метрика оцінювання результатів науково-освітньої діяльності, що має грошовий еквівалент. Конструктивно, всі позитивні досягнення вчених та підрозділів повинні мати свої соціально значущі коефіцієнти матриці компетенцій чи рейтингових показників. Тоді вчені будуть справді займатися наукою, не побоюючись, що реальним результатам керівництво віддасть перевагу паперовим звітам та договорам.

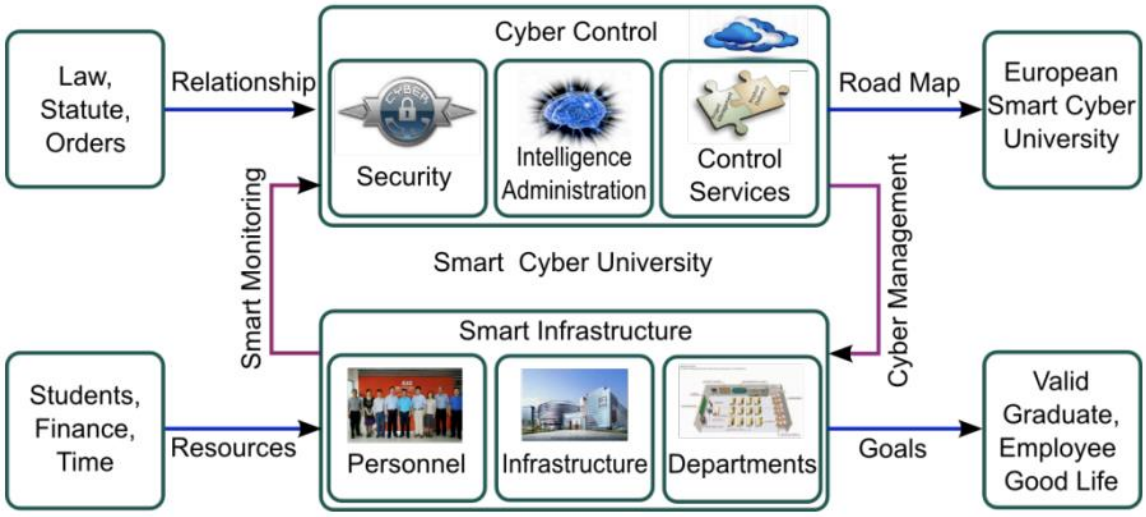

Рисунок 1.2 – Cyber Physical System – Smart Cyber University

Метрика – спосіб вимірювання відстані між об'єктами, процесами або явищами шляхом порівняння їхніх параметрів. Якість – сукупність властивостей об'єкта (суб'єкта, процесу чи явища), що зумовлюють його придатність задовольняти певні потреби відповідно до призначення. Кіберфізична система Smart Cyber University (SCU) – сукупність взаємопов'язаних компонентів: 1) online моніторингу науково-освітніх процесів та 2) управління ними шляхом використання матеріально-технічної, методичної та організаційної інфраструктури, відносин заслуженого морального та матеріального стимулювання співробітників, які забезпечують випуск придатних для ринку спеціалістів. Світова спільнота практично перебуває у трьох десятиліття від створення штучного розуму людства. Тому сьогодні слід говорити про впровадження кіберсистеми – ідеального віртуального менеджера, непідкупного та неупередженого, толерантного та гуманного, справедливо управляючого з морально-етичних та юридичним нормам, виробленим та вистражданим людством. Ресурси управління не створюють продукцію, вони є «паразитуючої» частиною виробничо-орієнтованої системи разом з інфраструктурою сервісного обслуговування, які негативно впливають вартість кінцевої продукції. Але без управління та інфраструктури не можна створювати продукцію – виникне хаос. Тому будь-яка виробнича система повинна мати на меті – зменшити співвідношення між кадровою потужністю компонентів управління та виконання. Приватний бізнес успішно вирішує цю проблему, де метрика «час – гроші – якість» регулює чисельність управлінських кадрів та допоміжних ресурсів сервісного обслуговування на рівні не більше 20 відсотків.

Інноваційні сервіси, що формують розумний кіберуніверситет як структурний прототип глобального науково-освітнього віртуального кібер-простору Global Smart Cyber University. Хмарний кібер-сервіс захищеного електронного документообігу для цифрового моніторингу та інтелектуального кібер-управління науково-просвітницький процесами (створення, реалізація та утилізація документа), в форматі замкнутого циклу:

«факт - вимір - оцінка - дія» , повністю виключає паперові носії шляхом використання Cloud-Mobile Service Computing баз даних, цифрового підпису, ID-card, пошти та мобільного телефону.

Хмарний кібер-сервіс надання освітніх послуг у вигляді MOOC online та onsite курсів, а також управління освітнім процесом на основі прозорого розподілу фінансових та часових (кредитних) ресурсів між підрозділами та співробітниками в строгому відповідно до метричних оцінювань вкладі кожного суб'єкта в актив та імідж університету.

Хмарний кібер-сервіс моніторингу та управління науково-освітнім процесом студента в реальному масштабі часу, генерування та зберігання електронних документів для його супроводу у часі та просторі через створення персонального віртуального Кабінету, пов'язаного з мобільним пристроєм та e-mail (рис.1.3).

Рисунок 1.3 – Інноваційні сервіси розумного кібер-університету

Кіберфізична система моніторингу та управління результативно та виграшно масштабується практично на всі сфери людської діяльності, пов'язані з економікою, політикою, соціологією, наукою, освітою, енергетикою, охороною здоров'я, управлінням суспільством, ресурсами, транспортними засобами Кіберфізичні системи ототожнюються зі

створенням "розумних" фабрик, університетів, будинків, міст та країн, критичних інфраструктур, захистом інформації та приватної власності, управління авіацією та космонавтикою. Всі перераховане є неповний перелік найактуальніших, на ринку, провідних компаній та університетів світу, питань кіберуправління фізичними об'єктами та процесами.

В сервісі управління навчанням студента використовуються такі документи: освітня програма, навчальний план, робочий навчальний план, робоча програма навчальної дисципліни, графік навчального процесу, розклад занять.

Навчальний план складається на підставі відповідної освітньої програми і визначає графік навчального процесу, форми організації освітнього процесу, перелік, обсяг та логічну послідовність освітніх компонентів, види та обсяг навчальних занять, форми поточного і підсумкового контролю.

Графік навчального процесу визначає календарні терміни теоретичної та практичної підготовки, підсумкового контролю (екзаменаційних сесій), підготовки та проведення атестації, канікул.

Розклад занять має забезпечити виконання робочого навчального плану на кожний семестр, враховуючи структурно-логічну схему підготовки у даному семестрі та контрольні заходи в повному обсязі щодо навчальних занять.

1.2 Облік успішності виконання завдання

Успіх виконання завдань та навчання залежить від різних факторів. Насамперед, від оцінок за роботи, вчасного їх виконання та завершення виконання по факту. Оцінка успіху в мобільному застосунку – це автоматичний процес. Користувач виконує завдання, а програмне забезпечення демонструє результат та оцінку виконання.

Аналізу виконання графіку навчального процесу моделюється на основі показників робочої програми навчальної дисципліни (рис. 1.4).

<table>
<tr><td rowspan="2">Найменування показників</td><td colspan="2">Характеристика навчальної дисципліни*</td></tr>
<tr><td>денна форма навчання</td><td>заочна форма навчання</td></tr>
<tr><td>Кількість кредитів ЄКТС*4,0</td><td colspan="2">вибіркова</td></tr>
<tr><td>Модулів** ___1__</td><td colspan="2">Рік підготовки:</td></tr>
<tr><td>Змістових модулів __1___</td><td>__1-й</td><td>__-й</td></tr>
<tr><td rowspan="11">Загальна кількість годин*
_____120_____

Мова навчання
___українська_____</td><td colspan="2">Семестр</td></tr>
<tr><td>___2-й</td><td>___-й</td></tr>
<tr><td colspan="2">Кількість годин</td></tr>
<tr><td>48</td><td></td></tr>
<tr><td colspan="2">Навчальні заняття:
1) лекції, год</td></tr>
<tr><td>24</td><td></td></tr>
<tr><td colspan="2">2) практичні, год</td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="2">3) лабораторні, год</td></tr>
<tr><td>16</td><td></td></tr>
<tr><td colspan="2">4) консультації, год</td></tr>
<tr><td>8</td><td></td></tr>
<tr><td colspan="2">Самостійна робота, год</td></tr>
<tr><td>72</td><td></td></tr>
<tr><td colspan="2">2) курсова робота(проект), год</td></tr>
<tr><td>–</td><td></td></tr>
<tr><td colspan="2">Вид контролю: екзамен
_____</td></tr>
</table>

Рисунок 1.4 – Приклад опису навчальної дисципліни МПЯ ПЗ у робочій програмі

Розклад занять (рис. 1.5) у вигляді .pdf файлів або зображень незручно оновлювати та не має можливості переглянути детальнішу інформацію о навчальних дисциплінах.

19	20	21	22
06.06.2022	13.06.2022	20.06.2022	27.06.2022
07.06.2022	14.06.2022	21.06.2022	28.06.2022
МПЯПЗ Лк ФІЛІЯ СКСм-21-1			
08.06.2022	15.06.2022	22.06.2022	29.06.2022
09.06.2022	16.06.2022	23.06.2022	30.06.2022
10.06.2022	17.06.2022	24.06.2022	01.07.2022
	МПЯПЗ Лб ФІЛІЯ СКСм-21-1		МПЯПЗ Лб ФІЛІЯ СКСм-21-1
	МПЯПЗ Лб ФІЛІЯ СКСм-21-1		МПЯПЗ Лб ФІЛІЯ СКСм-21-1
11.06.2022	18.06.2022	25.06.2022	02.07.2022
12.06.2022	19.06.2022	26.06.2022	03.07.2022

Рисунок 1.5 – Фрагмент розкладу навчальної дисципліни МПЯ ПЗ

Реалізація графіку навчального процесу (рис. 1.6), як сервіс розумного кіберуніверситету, значно покращує зручність користуванням розкладом.

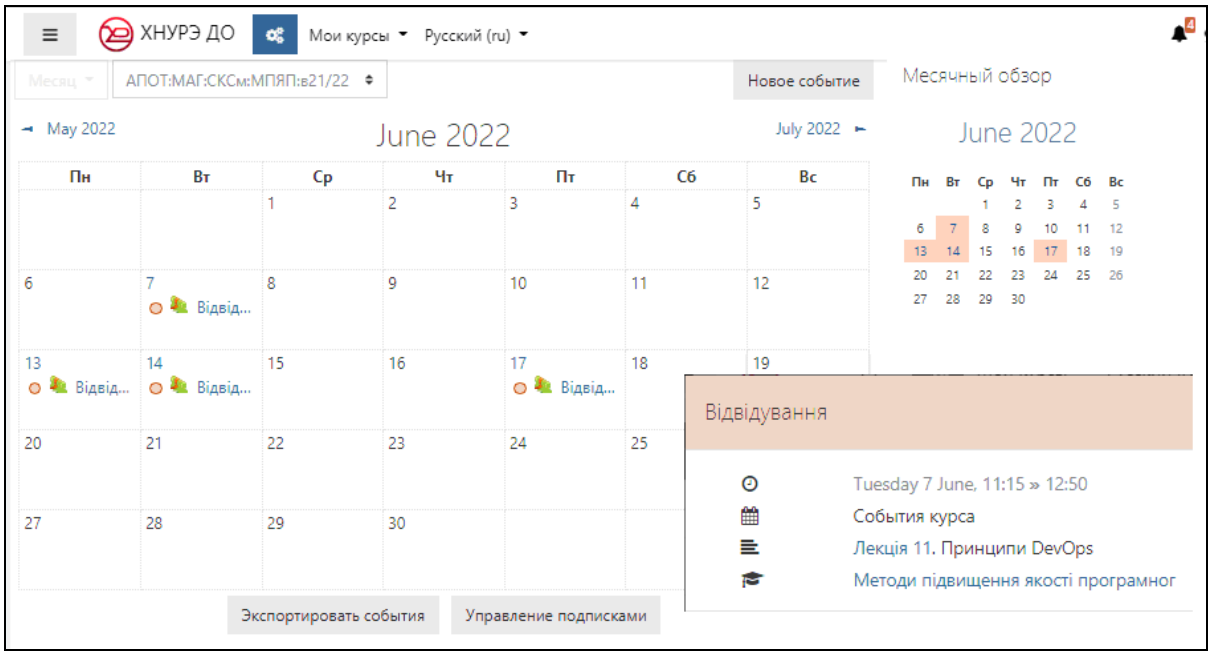

Рисунок 1.6 – Графік навчального процесу в dl.nure

В сучасному житті існують подібні мобільні застосунки, які допомагають користувачам контролювати себе та свій графік. Серед них популярність мають застосунки для відстеження за харчуванням. Наприклад, Lifesum (рис.1.7). Це програма, яка допоможе перейти на здоровий спосіб

життя, почати правильно харчуватися і скинути вагу. Вона допомагає стежити за водним балансом, кількістю калорій на добу, складом раціону та багато іншого.

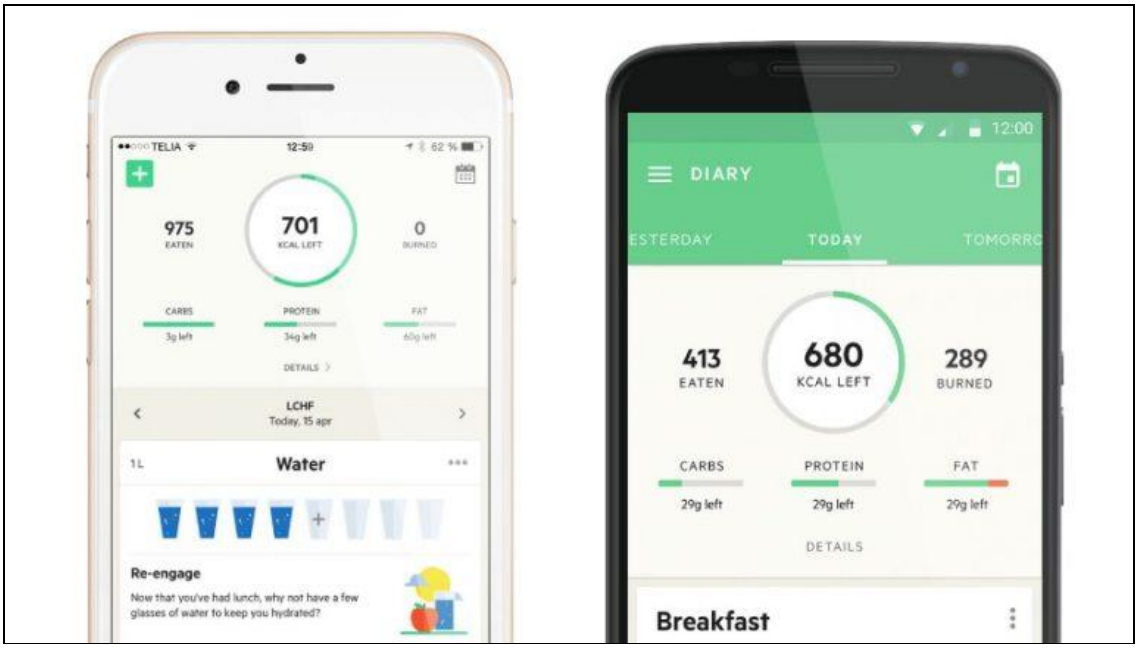

Рисунок 1.7 – Інтерфейс Lifesum

Розробники програми пішли далі банального підсумовування калорій і пропонують виходячи з фізіологічних даних, віку і ваги підібрати певний план харчування. Проте людський організм – це не машина, а мобільний застосунок не може замінити дієтолога. Дана програма корисна як калькулятор, що дисциплінує.

Подібна мета ставиться перед програмним забезпеченням у даній роботі. Застосунок не буде заміняти викладача чи виконувати його роботу. Він програмується для покращення та полегшення навчання студентів.

1.3 Електронне навчання в освітньому сегменті

Світова промисловість електронного навчання ще 2000 року становила 48 млрд доларів. Електронне навчання виникло завдяки розвитку інтернету

та мультимедіа, ключовими моментами є консалтинг, контент, технології, послуги та підтримка.

Стрімкість сучасного світу вимагає застосування найшвидших і найдешевших способів процесів генерації та передачі знань. Електронне навчання як інструмент відповідає цим цілям.

За даними Babson Survey Research Group, у 2012 році в США в онлайн-навчання у вищих навчальних закладах було залучено 6,7 мільйонів студентів. Онлайн-освіта швидко розвивається й у провідних дослідницьких інститутах навіть розроблено докторські програми, представлені онлайн. Багато вищих навчальних закладів, інститутів на комерційній основі пропонують зараз навчання в онлайн-класах. Кількість таких навчальних закладів зростає у міру розвитку та здешевлення технологій електронного навчання. Також потрібно враховувати, що для роботи зі студентами в режимі онлайн навчальним закладам потрібен кваліфікований персонал, який володіє комп'ютером та інтернет-технологіями.

Електронне навчання щодо нове явище, тому воно стикається з низкою проблем у розвитку, не тільки в технічній частині, а й на боці законодавчої бази, стандартизації та ін. До основних проблем розвитку електронного навчання можна віднести:

- відсутність критеріїв та загальних стандартів якості електронних навчальних матеріалів, недостатньо розкрито потенціал можливості передачі інформації по Інтернету: в основному використовуються такі форми як текст і проста графіка;

- правові проблеми, пов'язані як з нормативно-правовим забезпеченням електронного навчання, так і з питаннями захисту авторських прав;

- питання фінансування (витрати на розробку, зберігання даних, створення веб-ресурсів, їх підтримку діяльності та оновлення);

- кадрові проблеми (брак кваліфікованого персоналу, складність його навчання, оскільки одночасно потрібно охопити і предметну область, і застосування IT-технологій, і художнього оформлення матеріалів).

1.4 Moodle

Moodle (Modular Object-Oriented Dynamic Learning Environment) – це безкоштовна, відкрита (Open Source) система управління навчанням. Вона реалізує філософію «педагогіки соціального конструктивізму» та орієнтована насамперед на організацію взаємодії між викладачем та учнями, хоча підходить і для організації традиційних дистанційних курсів, а також підтримки очного навчання.

Типовий сучасний вищий навчальний заклад автоматизує свою діяльність через велику кількість взаємопов'язаних програмних систем для різних груп користувачів. Через таке різноманіття установи змушені керувати складним програмним комплексом в гетерогенному середовищі, який включає в себе нові та застарілі інформаційні системи, які вимагають обміну даними між собою та можуть бути побудованими на різних операційних системах, базах даних, тощо.

Moodle вже має модулі інтеграції з різними системами. Але чи можливо створити загальну систему контролю навчання тільки засобами Moodle. Так, можна створити просту але ефективну систему. І нові версії Moodle дають нам для цього більше можливостей

Moodle має широкий набір функціональності, притаманний платформам електронних систем навчання, системам управління курсами (CMS), системам управління навчанням (LMS) або віртуальним навчальним середовищам (VLE). Moodle надає можливість викладачам створювати ефективні сайти для онлайн-навчання. Moodle можна використовувати як в навчанні школярів, студентів, так і при підвищенні кваліфікації, бізнес-навчанні.

Типова функціональність Moodle включає:

- здача завдань;
- дискусійні форуми;

– завантаження файлів;

– оцінювання;

– обмін повідомленнями;

– календар подій;

– новини та анонси подій (для різних рівнів: сайт, курс, навчальна група);

– онлайн тестування;

– вікі.

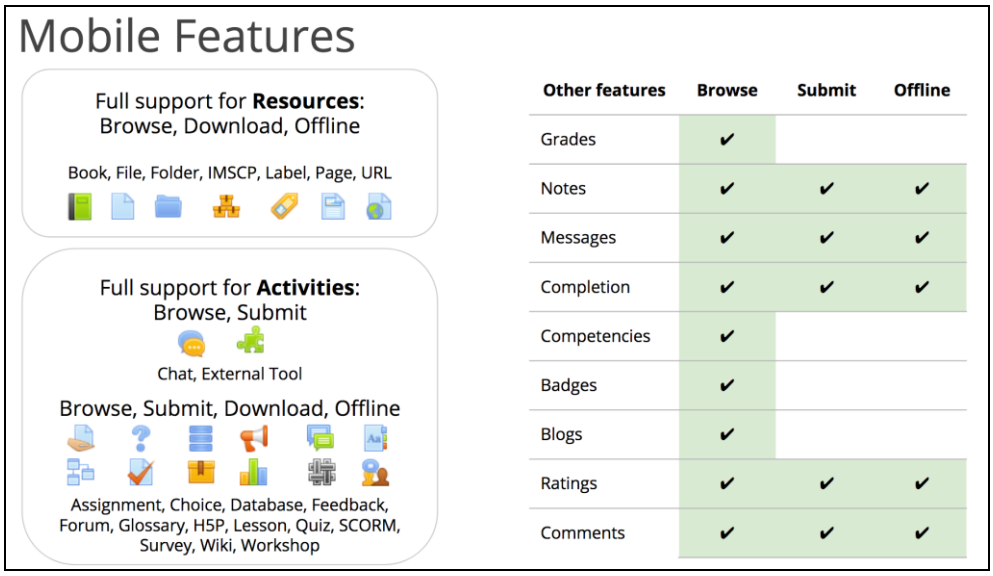

Рисунок 1.8 – Функціональність Moodle

В категорії кафедр всі курси поділяються за конкретними кафедрами (а при потребі поділ може бути поглиблено на освітньо-кваліфікаційні рівні). Ці курси представляють собою звичайні електронні навчальні курси. Обов’язковою умовою для них є включення способу зарахування через мета-курси.

Детальніше розглядаємо категорію факультети. Там створюються категорії під кожен факультет. В кожному факультеті на базі елементу “курс” створюються навчальні плани для кожної спеціальності та терміну навчання.

Наприклад, “Економічна кібернетика. 2013-2017” (рік вступу та рік закінчення). Такі курси будуть багатофункціональними ресурсними системами як для студентів, так і для працівників деканату.

Ще одним важливим питанням при роботі з користувачами мета-курсів є їх розподіл на групи. Проблематичність вирішення цього питання у попередніх версіях Moodle не дозволяла ефективно застосувати подібну практику. Але тепер маємо рішення. Для цього використовуємо глобальні групи moodle (гурти, когорти). Вони можуть бути розміщені як на найвищому рівні сайту, так і на рівнях категорій. Використовуючи таку можливість створюємо гурти студентів на рівні категорій їх факультетів. В цих же категоріях надаємо ролі менеджерів для працівників деканату, щоб вони могли працювати з курсами-потоками. В їх обов’язки буде входити:

- створення курсів-потоків для різних спеціальностей та років вступу;
- запис до них відповідних студентів;
- ведення курсів-потоків в межах налаштування дат та параметрів доступу;
- електронне інформування студентів потоку через новини та оголошення.

Студенти до своїх курсів-потоків записуватимуться через метод зарахування “Синхронізація гурту”. Таким чином переведення студента до певного гурту автоматично синхронізує його з курсом-потоком та всіма навчальними курсами поточного семестру для цього потоку. В процесі синхронізації гурту з курсом-потоком необхідно прив’язати його до такої ж групи в цьому курсі. В результаті працівники деканату отримують можливість переглядати успішність студентів за окремими групами.

Рисунок 1.9 – Ролі та їх функціональні обов'язки

Для того ж, щоб ці групи були автоматично створені і в мета-курсах, з автоматичним зарахуванням в них відповідних студентів, потрібно встановити ще один додатковий модуль - Meta-course group synchronization. Оскільки на даний час в мета-курсах відсутня можливість синхронізації груп з дочірніми курсами, цей модуль додає таку функціональність шляхом прослуховування подій групових заходів в дочірніх курсах з подальшим оновленням груп в мета-курсах із зарахуванням до них студентів.

Як результат, отримуємо налагоджену систему зарахування студентів в електронні навчальні курси з розподілом по групах. При чому, якщо якогось студента зараховують на навчання на факультет пізніше, його достатньо ввести до відповідного гурту, далі він буде автоматично зарахованим до всіх потрібних курсів. Аналогічно, відрахованого студента прибираємо з гурту, і він перестає бути зарахованим на всіх курсах.

В результаті правильного налаштування та використання модулів системи Moodle можна побудувати просту але функціональну систему навчання та контролю за ним з боку деканату (рис.1.10). З автоматичним зарахуванням студентів на потрібні курси та автоматичним збором

отриманих ними під час навчання балів. Важливим є відносно легке та коректне вбудування отриманої системи в загальне навчально-інформаційне середовище університету.

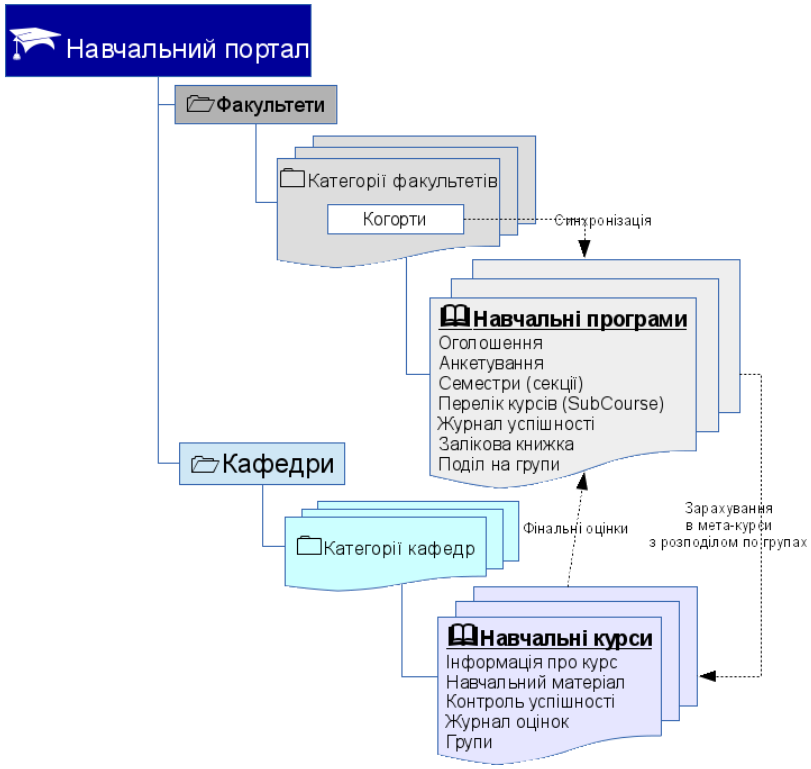

Рисунок 1.10 – Схема функціонування системи контролю за навчанням

Є кілька способів відстежувати прогрес студента в Moodle. Кожен курс має власний Журнал оцінок, доступ до якого можна отримати в меню Адміністрування курсу > Налаштування журналу оцінок. Деякі дії, як завдання та тести, повертають оцінки до цього журналу оцінок. Також вчителі можуть вносити оцінки безпосередньо в журнал оцінок.

Компетенції описують рівень розуміння або володіння учнем певними предметними навичками. Освіта на основі компетенцій (CBE), також відома як навчання на основі компетенцій або навчання на основі навичок, відноситься до систем оцінювання та виставлення оцінок, у яких учні демонструють ці компетенції.

Як розширення завершення діяльності, увімкнення завершення курсу дозволяє офіційно позначити курс як завершений вручну або автоматично відповідно до визначених критеріїв. Якщо додано блок статусу завершення курсу, студенти зможуть бачити свій прогрес протягом курсу. Викладачі можуть переглядати загальний прогрес студентів на шляху до завершення курсу в Адміністрування курсу>Звіти>Завершення курсу.

Рисунок 1.11 – Вигляд завершення курсу для студента

Criteria	Announcements from your tutor	Prior Knowledge assessment	Factual recall test	Useful links	Video resources	Course discussion	Group Project	Reflective journal	Course complete
First name / Surname									
Frances Banks	☑	☑	☑	☐	☐	☑	☐	☐	☐
Mark Ellis	☑	☑	☑	☑	☑	☐	☐	☐	☐
Brian Franklin	☐	☑	☑	☐	☐	☑	☑	☐	☐
Barbara Gardner	☑	☑	☑	☑	☐	☑	☐	☐	☐
Amanda Hamilton	☑	☑	☑	☑	☑	☑	☑	☑	☑

Рисунок 1.12 – Вигляд завершення курсу для викладача

Деякі звіти про курси доступні для вчителя на його курсі, щоб допомогти йому відстежувати прогрес своїх студентів. На додаток до згаданих вище звітів про діяльність і завершення курсу (які доступні, лише якщо ці налаштування ввімкнені), є також звіти про діяльність, звіти про участь і загальні журнали курсу.

1.5 Інструменти розробки

Перед тим як створювати мобільний застосунок чи навіть розглядати його майбутню структуру, необхідно вирішити низку запитань та визначитися із платформою, на якій буде використовуватися забезпечення. Після цього обрати відповідну мову програмування серед множини існуючих мов та середовище розробки, як робочий простір розробника.

1.5.1 Мова програмування

Оскільки розробка програмного забезпечення виконується під платформу IOS, то як мову програмування було обрано SWIFT. Це швидка і ефективна мова програмування з відгуком в реальному часі, яку легко можна вставити в готовий код Objective-C. Тепер розробники можуть не тільки писати більш надійні і безпечні коди, але і економити час та створювати додатки з розширеними можливостями[5].

Потенціал Swift оцінили навчальні заклади по всьому світу. Вони використовують Swift і Xcode на Mac для навчання, забезпечуючи своїх студентів найкращими інструментами для створення додатків. А з безкоштовною навчальною програмою від Apple «Розробка на мові Swift» навіть новачки легко переходять від основ програмування до професійної розробці.

Swift безкоштовно доступний для використання розробниками, викладачами і студентами за ліцензією на поширення ПЗ з відкритим вихідним кодом Apache 2.0. Ми надаємо виконавчі файли для OS X і Linux,

які дозволяють компілювати код для iOS, OS X, watchOS, tvOS і Linux. А щоб прискорити розвиток і перетворення Swift в ще більш потужний мову, ми створили нове співтовариство, де користувачі можуть безпосередньо вносити свій вклад в вихідний код Swift.

1.5.2 Середовище розробки

Оскільки мовою програмування обрано SWIFT, то відповідно необхідно обрати XCode як середовище розробки.

Xcode складається з набору інструментів, які розробники використовують для створення додатків для платформ Apple. Використання Xcode необхідне для керування всім робочим процесом розробки: від створення програми до тестування, оптимізації та надсилання її в AppStore.

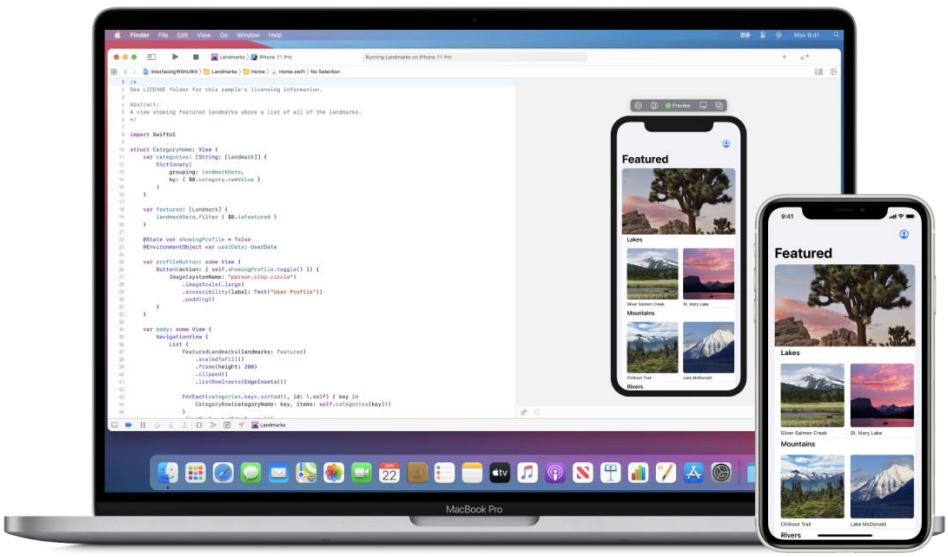

Рисунок 1.13 – Знімок екрана MacBookPro під управлінням Xcode

Xcode надає Simulator для швидкого створення прототипів і тестування програми в імітованому середовищі, коли реальний пристрій недоступний. Simulator забезпечує середовища для пристроїв iPhone, iPad, AppleWatch та Apple TV з різними налаштуваннями, файлами та версіями операційної системи.

Середовище містить інструменти для профілювання та аналізу програми, підвищення продуктивності та пошуку проблем із пам’яттю.

1.6 Постановка завдання

В структурі кіберуніверситету основним є хмарний кібер-сервіс моніторингу та управління науково-освітнім процесом студента в реальному масштабі часу. Система управління навчальним процесом університету складається з бази навчальних планів, графіків навчального процесу, розкладу занять відомостей отриманих оцінок тощо.

Об’єкт дослідження у роботі – система управління навчальним процесом у складі кіберуніверситету.

Предмет дослідження – моделі, методи та процедури оптимізації навчального процесу студентів з використанням мобільних технологій.

Мета роботи – розробка мобільного застосунку в рамках реалізації хмарних сервісів кіберуніверситету для обліку успішності виконання навчального графіку студента з відображенням шкали прогресу у відповідності до моделі навчального процесу, яка складається з таких компонентів:

- відвідування;
- вивчення матеріалу;
- виконання завдань;
- вчасності виконання завдань;
- перевірки знань;
- оцінювання.

Мобільний застосунок має містити список курсів, функцію додавання дедлайнів, відображення прогресу навчання за результатами виконання навчальних завдань.

2 РОЗРОБКА ПРОГРАМНОЇ ЧАСТИНИ ПРОЕКТУ

2.1 Загальна архітектура системи

При виборі архітектури було визначено такі основні вимоги:

- збалансований розподіл обов'язків між сутностями із жорсткими ролями;
- тестованість;
- простота використання та низька вартість обслуговування.

Базуючись на визначених вимогах була обрана модульная архітектура Model View Presenter (MVP) для мобільного додатку. Мобільний застосунок має мати у своїй складовій декілька незалежних один від одного модулів.

Модель - відповідає за використання предметних (бізнес, domain) даних. Це також може бути data access layer. Моделі бувають двох видів: активні та пасивні. Активні моделі вміють повідомляти оточуючих про зміни у собі, а пасивні – ні.

Вид (View) - відповідає за шар представлення (GUI). Вигляд не обов'язково повинен бути пов'язаний з UI малюванням: представлення данних у вигляді діодів, що змінюють один одного на arduino-платі — цим все займатиметься View. Крім представлення користувачеві даних, View має ще одне важливе завдання: прийняти подію користувача (натискання на кнопку, наприклад).

Презентер – відповідає за зв'язок моделі з контролером. Здебільшого займається тим, що прокидає події моделі на вигляд, а події виду – на модель, відповідним чином їх перетворюючи та обробляючи. Наприклад, змінюючи модель у відповідь на дії користувача на екрані, або змінюють вигляд у відповідь на зміни моделі. Зазвичай є пасивними учасниками процесу і реагують лише зовнішні стимули (події від View чи Model).

Наступна структура ізолює модуль логіки програми від модулів

отримання та відображення інформації. Так як реалізація мобільного додатку на платформу iOS має на увазі використання мови програмування Swift або Objective-C, то реалізація взаємодії між ними буде використовувати протокольно-орієнтоване програмування, що виключить будь-які можливі залежності між модулями. Протоколи будуть описувати необхідні інтерфейси, що потрібно реалізувати розробнику конкретної програми. Модулі будуть працювати лише з відомим їм інтерфейсами, що виключить їх залежність від конкретної реалізації клієнтського додатку

2.2 База даних

Для зберігання даних у застосунку використовується UserDefaults. UserDefaults – інтерфейс до бази даних користувача за замовчуванням, де постійно зберігаються пари ключ-значення під час запуску програми. UserDefaults здатний підтримувати всі основні типи даних: Bool, Dictionary, Int, String, Data, and Array. Але не підтримує опціональні типи.

Клас UserDefaults забезпечує програмний інтерфейс для взаємодії з системою за замовчуванням. Система за замовчуванням дозволяє програмі налаштовувати свою поведінку відповідно до вподобань користувача. Програма зберігає параметри, призначаючи значення набору параметрів у базі даних користувача за замовчуванням. Параметри називаються параметрами за замовчуванням, оскільки вони зазвичай використовуються для визначення стану програми за замовчуванням під час запуску або того, як вона діє за замовчуванням.

Під час виконання використовуються об’єкти UserDefaults, щоб читати параметри за замовчуванням, які використовує програма, із бази даних за замовчуванням користувача. UserDefaults кешує інформацію, щоб уникнути необхідності відкривати базу даних користувача за замовчуванням кожного разу, коли потрібно значення за замовчуванням. Коли встановлюється значення за замовчуванням, воно змінюється синхронно у процесі та

асинхронно для постійного сховища й інших процесів

Важливо не намагатись напряму отримати доступ до підсистеми налаштувань. Змінення файлів списку властивостей налаштувань може призвести до втрати змін, затримки відображення змін і збоїв програми. Щоб налаштувати параметри, натомість існує утиліта командного рядка за замовчуванням у macOS.

2.3 Модель навчального процесу

Модель навчального процесу – це еталонне уявлення про навчання студентів, його конструювання в умовах конкретних освітніх програм. Вона визначає цілі, основи організації та проведення навчального процесу. Основною метою навчання є оволодіння знаннями, уміннями, навичками, відібраними на основі принципу науковості, системності, доступності, наочності.

В основі цих моделей лежить уявлення про навчання, як про інформаційний процес, що полягає у сприйманні, збереженні та відтворенні та переробці наукової інформації. На цій основі якість засвоєння визначається трьома показниками :

- повнотою відтворених знань;
- їх використанням за зразком;
- використанням в нестандартних ситуаціях.

Відповідно до головних критеріїв до навчання, модель навчального процесу складається з таких компонентів:

- відвідування;
- вивчення матеріалу;
- виконання завдань;
- вчасності виконання завдань;
- перевірки знань;
- оцінювання.

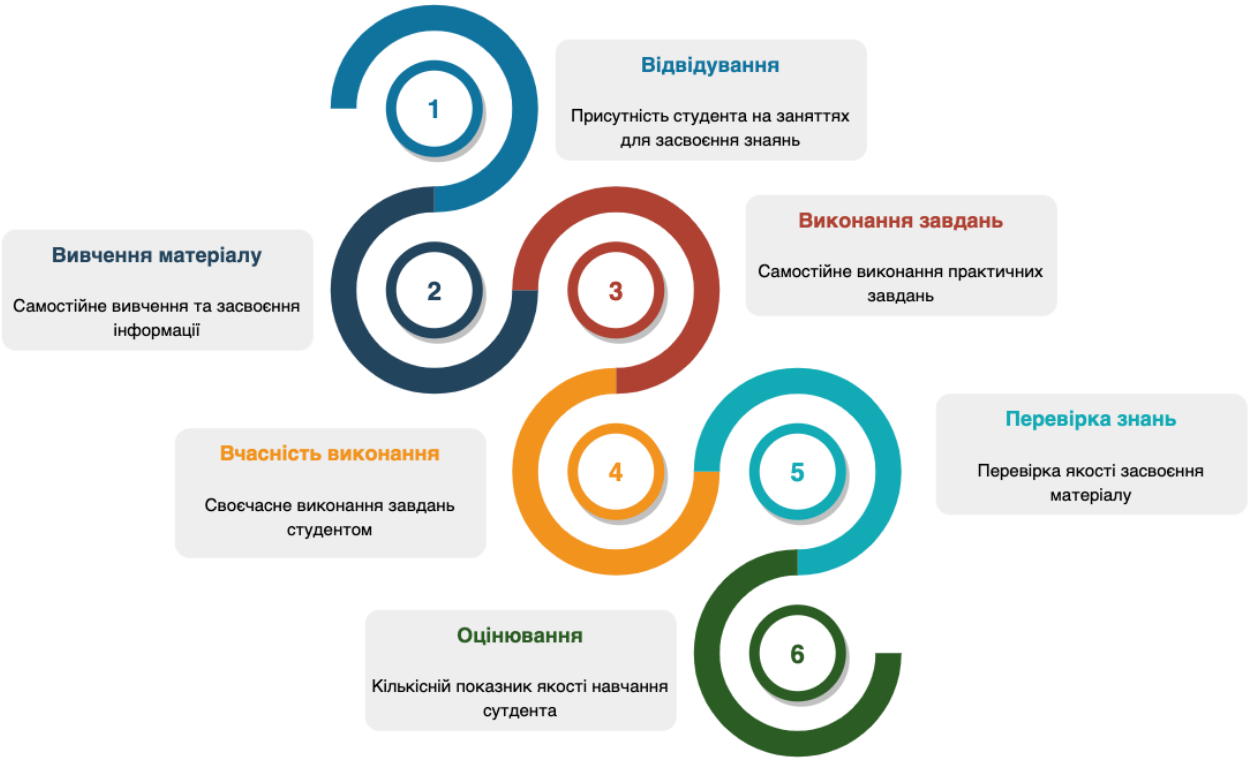

Рисунок 2.1 – Модель навчального процесу

Саму у такій послідовності формується розуміння о навчальному процесі. Студенти отримують чітке уявлення про необхідну послідовність дій яка потрібна для здобуття знань. На основі цієї інформації вчитель має здатність якісно оцінити здобуті знання студента з певного предмету.

Рисунок 2.2 – Модель відвідування

Відвідування це перша складова моделі навчання. Вчитель формує модель відвідування (рис2.2) індивідуально для кожного предмету. Головні заходи для відвідування студентами:

– лекції;
– лабораторні роботи;
– практичні роботи.

Рисунок 2.3 – Модель виконання завдань

На основі засвоєння інформації при відвідуванні занять, студенти повинні виконати певні завдання для перевірки знань. Вчитель формує модель виконання завдань (рис 2.3) на основі навичок які потрібно здобути студентам.

Модель оцінювання якості знань студента формується на основі таких показників:

– відвідування;
– виконання завдань;
– вчасність виконання завдань.

Рисунок 2.4 – Модель оцінювання

2.4 Структура головного екрану

Головний екран має назву «MyDeadlines» (рис.2.5). На ньому відображено перелік предметів у формі карток з прогресом виконання навчання. Картка буде реалізована за допомогою класу UITableViewCell. Це візуальне відображення окремого рядка у вигляді таблиці. На картці зображено текст з назвою предмета, текст з кількістю завдань та процентну шкалу прогресу.

Лістинг 2.1 – Клас візуального відображення окремого рядка у таблиці

```
final class TaskCell: UITableViewCell {

    @IBOutlet private weak var nameLabel: UILabel!
    @IBOutlet private weak var tasksCount: UILabel!
    @IBOutlet private weak var progressView: UIProgressView!
    @IBOutlet private weak var percentLabel: UILabel!
```

```
    @IBOutlet private weak var shadowView: UIView!

    override func awakeFromNib() {
        super.awakeFromNib()

        setupViewAppearance()
    }
}
```

Клас описує усі візуальні компоненти рядка (рис.2.5) для відображення статусу задачі.

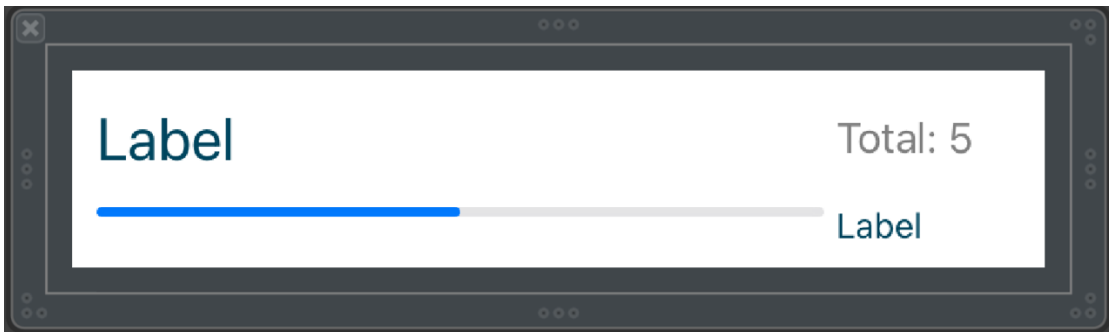

Рисунок 2.5 – Зовнішній вигляд рядка TaskCell

Клас TaskCell містить 3 функції.

Лістинг 2.2 – Функція оновлення рядка TaskCell

```
func update(with task: TaskModel) {
    nameLabel.text = task.name
    tasksCount.text = "total: \(task.lessons.count)"

    getProgres(task: task)
}
```

Функція оновлює текстові компоненти (UILabel) та оновлює показники прогресу за допомогою функції getProgres.

Лістинг 2.3 – Функція оновлення прогресу

```
func getProgres(task: TaskModel) {
    var finished: Int = 0
    let total = task.lessons.count
    for item in 0..<total {
        if task.lessons[item].isFinished == true {
            finished += 1
```

```
        }
    }
    var formula: Float = 0.0

    if finished == total {
        formula = 1.0
    } else {
        formula = (Float((100/total) * finished) / 100)
    }
    progressView.progress = formula
    percentLabel.text = "\(Int(formula * 100))%"
    setColor(progress: formula)
}
```

У кожного показнику прогресу є відповідний колір, який задається функцією setColor.

Лістинг 2.4 – Функція оновлення кольору

```
func setColor(progress: Float) {
    if progress <= 0.3 {
        progressView.progressTintColor = .red
    } else if progress <= 0.5 {
        progressView.progressTintColor = .orange
    } else if progress <= 0.7 { // 0.8
        progressView.progressTintColor = .yellow
    } else if progress <= 1.0 {
        progressView.progressTintColor = .green
    }
}
```

Текстові мітки відображають статичний текст і реалізовані в класі UILabel. Процентна шкала створена за допомогою UIProgressView. На основі класу UITableView (рис2.6) реалізуємо показ карток в стовпці. UITableView – це відображення, в якому дані представлені в вигляді рядків, розташованих в одному стовпці.

Класс реалізує головні методи делегату UITableViewDelegate для взаємодії з таблицею.

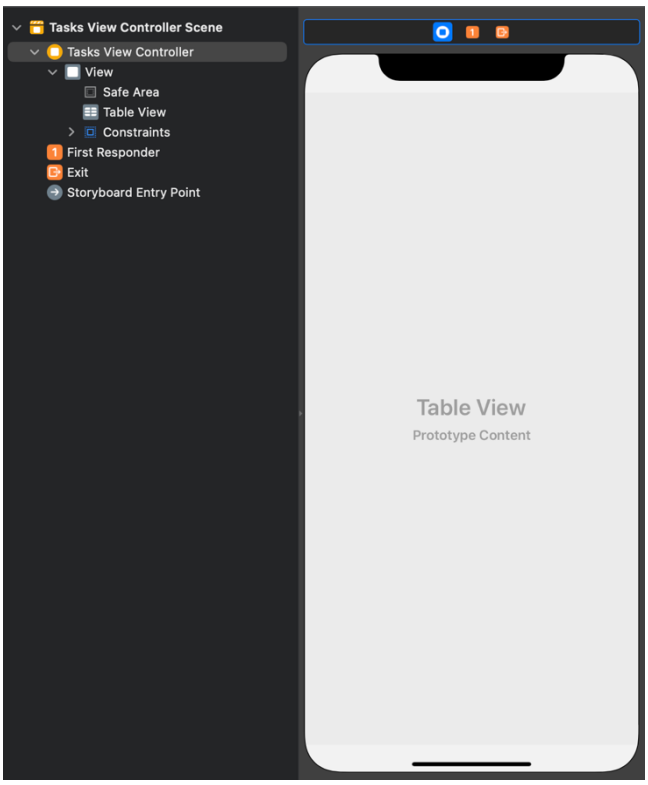

Рисунок 2.6 – Зовнішній вигляд UITableView

Лістинг 2.5 - Функція відображення кнопки AddNewTask

```
func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
    if presenter.tasks.count == 0 {
        setEmptyView()
    } else {
        self.tableView.backgroundView = nil
    }
    return presenter.tasks.count
}

func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
    let cell = tableView.dequeueReusableCell(withClass: TaskCell.self)
    cell.update(with: presenter.tasks[indexPath.row])
    cell.layer.shadowColor = UIColor.black.cgColor
    cell.layer.shadowOffset = CGSize(width: 0, height: 0)
    cell.layer.shadowOpacity = 0.6
    return cell
}
```

```
        func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
            presenter.onTaksInfo(index: indexPath.row)
        }

        func tableView(_ tableView: UITableView, commit editingStyle: UITableViewCell.EditingStyle, forRowAt indexPath: IndexPath) {
            if editingStyle == .delete {
                presenter.removeItem(index: indexPath.row)
            }
        }
```

Для додавання нових карток предметів існує кнопка «Add». Реалізація кнопки створена за допомогою UIBarButtonItem. Це спеціальна кнопка для розміщення на панелі інструментів. На основі атрибута IBAction створено метод, який при натисканні на кнопку буде відкривати екран «AddNewTask».

Лістинг 2.6 - Функція відображення кнопки AddNewTask

```
private func setupNavigationBar() {
    navigationItem.title = "Tasks"
    let rightButton = UIBarButtonItem(
        title: "Add",
        style: .done,
        target: self,
        action: #selector(createNewTask(_:)))
    rightButton.tintColor = .white
    self.navigationItem.setRightBarButton(rightButton, animated: true)
}

@objc func createNewTask(_ sender: UIBarButtonItem) {
    presenter.addNewTask()
}
```

При натисканні на самі картки з основною інформацією про курс буде відкриватися новий екран «MyTask». На ньому буде показана повна інформація про завдання з предмету.

Рисунок 2.7 – Зовнішній вигляд головного екрану

2.5 Екран додавання предметів

Назва екрану «AddNewTask» (рис.2.8). На даному екрані буде відображено назву предмету, кількість лабораторних та кількість практичних занять. Для обрання цих параметрів використано текстові поля. Вони дозволяють користувачеві вводити текст в один рядок. Реалізація текстового поля здійснена в класі UITextField.

Лістинг 2.7 - Функція валідації тексту у полі UITextField

```
func validate() {
    if task.name == "" {
        view?.showAlert(title: "Please, enter task name", message: "")
    } else if labsCount == 0, pracCount == 0 {
        view?.showAlert(title: "Please, enter labs or practical quantity",  message: "")
```

```
        } else {
            if nameIsExclusive(nameField: task.name) {
                saveTask()
            }
        }
    }
```

Після заповнення текстових полів основною інформацією необхідно перейти до встановлення дедлайнів. Цей перехід реалізовано кнопкою «Next» за допомогою атрибута IBAction. Створено метод, який при натисканні на кнопку, відкриє екран «Setdeadlines».

Лістинг 2.8 – Інтерфейс модулю головної логіки екрану

```
protocol CreateTaskPresenter {

    var task: TaskModel { get }
    var name: String { get set }
    var dates: [Date] { get set }
    var labsCount: Int { get set }
    var pracCount: Int { get set }
    func setTaskDetails(with taskModel: TaskModel)
    func saveTask()
    func updateTask()
    func validate()
}
```

Саме через цей інтерфейс відбувається спілкування модулів відображення та обробки даних.

Лістинг 2.9 – Функція встановлення залежностей між модулями екрана CreateTask

```
final class CreateTaskAssembly {}

// MARK: - Assembly
extension CreateTaskAssembly: Assembly {
```

```
    func assemble(container: Container) {
        container.register(CreateTaskViewController.self) { resolver in
            let view = R.storyboard.createTask.instantiateInitialViewController()!
            let presenter = resolver.resolve(CreateTaskPresenter.self, argument: view)!
            view.inject(presenter: presenter)
            return view
        }

        container.register(CreateTaskPresenter.self) { (resolver, view: CreateTaskViewController) in
            let flow = resolver ~> AppFlow.self
            let presenter = CreateTaskPresenterImp(view: view, flow: flow, dataManager: resolver~>)
            return presenter
        }
    }
}
```

Завяки классу Assembly є можливість додавати гнучкі залежності між модулями.

Лістинг 2.10 – Функція встановлення візуальної складової екрану

```
    func setupViewAppearance() {
        navigationItem.title = "Create new task"
        self.navigationController?.navigationBar.tintColor = .white

        tableView.delegate = self
        tableView.dataSource = self
        tableView.separatorStyle = UITableViewCell.SeparatorStyle.none
        tableView.register(nibWithCellClass: CreateTaskCell.self)

        labStepper.value = 0
        labStepper.minimumValue = 0

        pracStepper.value = 0
        pracStepper.minimumValue = 0

        labLabel.text = "Labs quantity: \(presenter.labsCount)"
        pracLabel.text = "Practical quantity: \(presenter.pracCount)"

        setShadow(for: createView)
```

```
        setShadow(for: stepperView)
    }
```

Функция задає первинні значення змінним та визначає головні стилі таблиці. Залежності між делегатами також визначені у функції setupViewAppearance.

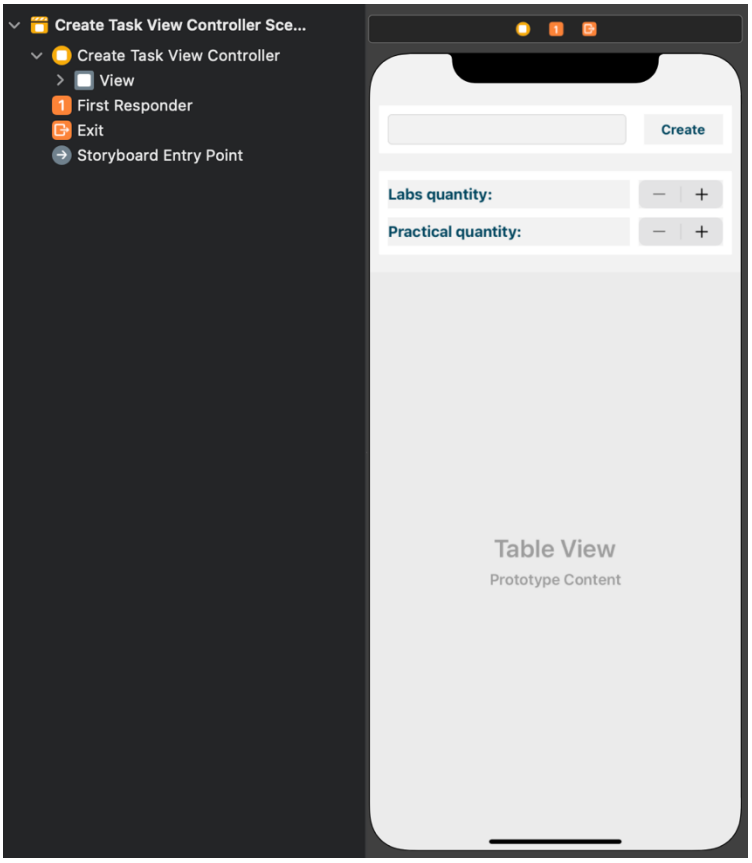

Рисунок 2.8 – Зовнішній вигляд екрану додавання предеметів

Наступна функція описує класс контроллера для додавання предметів.

Лістинг 2.11 – Функція встановлення візуальної складової екрану

```
final class CreateTaskViewController: UIViewController {

    @IBOutlet private weak var tableView: UITableView!
    @IBOutlet private weak var labStepper: UIStepper!
```

```
    @IBOutlet private weak var pracStepper: UIStepper!
    @IBOutlet private weak var nameField: UITextField!
    @IBOutlet private weak var labLabel: UILabel!
    @IBOutlet private weak var pracLabel: UILabel!
    @IBOutlet private weak var createView: UIView!
    @IBOutlet private weak var stepperView: UIView!


    private var presenter: CreateTaskPresenter!

    override func viewDidLoad() {
        super.viewDidLoad()

        setupViewAppearance()
    }

    func inject(presenter: CreateTaskPresenter!) {
        self.presenter = presenter
    }
}
```

2.6 Екран встановлення дедлайнів

На екрані «Setdeadlines» (рис.2.9) будуть відображатися: кількість лабораторних робіт і кількість практичних завдань, які були введені на екрані «Addnewtask» і для кожного завдання потрібно вибрати дедлайн. Вибір дедлайну реалізовано за допомогою класу UIDatePicker – це елемент управління для введення значень дати і часу.

Всі ці дані виведено на екран за допомогою класівUITableView і UITableViewCell. Кнопка «Finish» реалізована за допомогою атрибута IBAction. Створено метод, який при натисканні на цю кнопку буде відкривати екран «Mydeadlines» з доданою карткою.

Рисунок 2.9 – Зовнішній вигляд екрану дедлайнів

2.7 Екран конкретного предмета

На екрані «MyTask» (рис.2.10) буде відображатися повна інформація про завдання з обраного предмета на головному екрані. Виведення кількості лабораторних і практичних завдань та дедлайну зроблено за допомогою класу UILabel. Біля кожного завдання необхідне позначення, що робота виконана чи ні. Це реалізується за допомогою класу UIButton. Створено checkbox, який відзначатиме виконання завдання.

Лістинг 2.12 – Клас описує рядок з конкретною задачею

```
final class TaskInfoCell: UITableViewCell {

    @IBOutlet private weak var nameLabel: UILabel!
    @IBOutlet private weak var dateLabel: UILabel!
    @IBOutlet private weak var checkBox: UIButton!
    @IBOutlet private weak var shadowView: UIView!
```

```
    private var flag = false

    var callback : ((Bool) -> ())?

    override func awakeFromNib() {
        super.awakeFromNib()

        setupViewAppearance()
    }

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

Лістинг 2.13 – Функція оновлення рядка

```
    func update(with task: LessonModel) {
        nameLabel.text = task.name
        dateLabel.text = formatDate(date: task.deadline)
        flag = task.isFinished
        setCheckBox()
    }

    func setCheckBox() {
        if flag {
            checkBox.setImage(UIImage(named: "check"), for: .normal)
            callback?(flag)
            flag = false
        } else {
            checkBox.setImage(UIImage(named: "uncheck"), for: .normal)
            callback?(flag)
            flag = true
        }
    }
```

На екрані міститься процентна шкала, яка реалізована за допомогою класу UIProgressView. Клас UIProgressView надає властивості для управління стилем панелі прогресу, а також для отримання та встановлення значень,

закріплених за ходом виконання завдання [4].

Лістинг 2.14 – Головний інтерфейс комунікації модуля TaskInfo

```
protocol TaskInfoPresenter {

    var task: TaskModel { get set }
    var index: Int { get }
    func replace()
    func removeItem(lessonIndex: Int)
    func getOneTask()
}

extension TaskInfoPresenterImp: TaskInfoPresenter {

    func getOneTask() {
        guard let newTask = dataManager?.getOneTask(index: index) else { return }
        task = newTask
        self.view?.update()
    }

    func replace() {
        dataManager?.replaceTask(newTask: task, index: index)
    }

    func removeItem(lessonIndex: Int) {
        dataManager?.removeLesson(taskIndex: index, lessonIndex: lessonIndex)
        getOneTask()
    }
}
```

Данні функції обробляють усі можливі події від користувача які надходять з классу TaskInfoViewController.

Для повернення на головний екран створено кнопку «Back». Вона реалізована за допомогою атрибута IBAction. Керує створений метод, який при натисканні на цю кнопку, буде повертати користувача на головний екран «Mydeadlines».

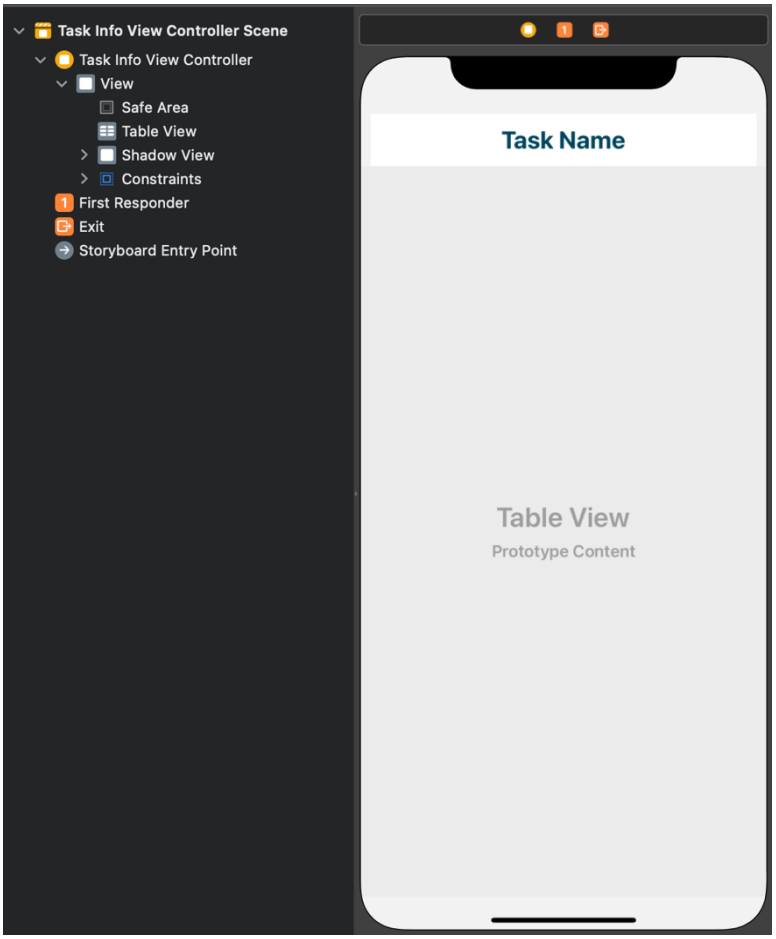

Рисунок 2.10 - Зовнішній вигляд екрану з завданнями

2.8 Екран останніх новин з сайту

На екрані News відображені останні новини з сайту кафедри. Усі новини відображені у вигляді таблиці (UITableView), яка я компонентом контролера NewsViewController.

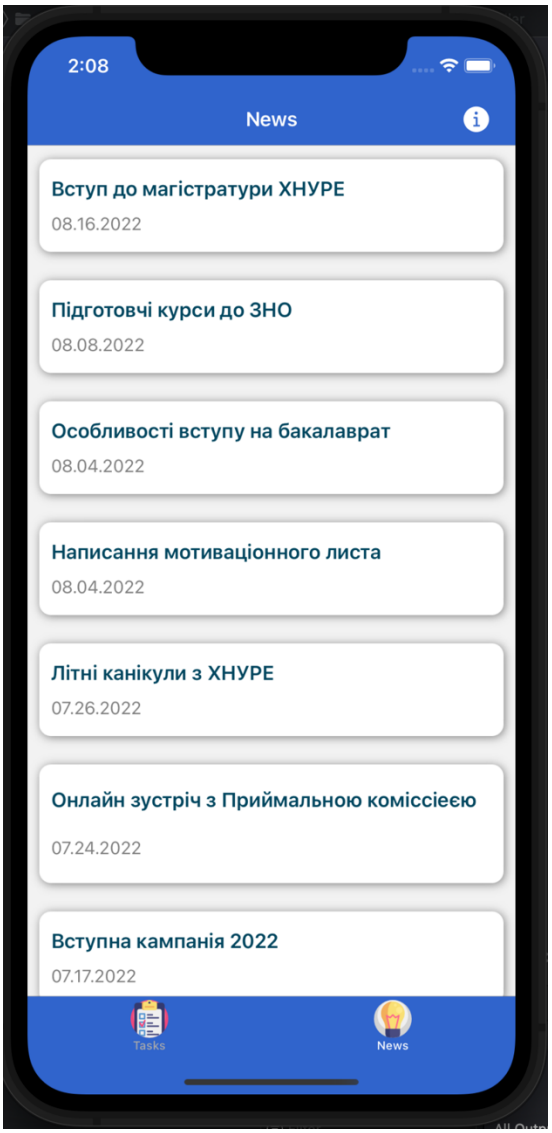

Рисунок 2.11 - Зовнішній вигляд екрану з новинами

Лістинг 2.15 – Реалізація методів UITableViewDelegate

```
// MARK: - UITableViewDelegate

extension NewsViewController: UITableViewDelegate, UITableViewDataSource {

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        presenter.posts.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withClass: PostTableViewCell.self)
```

```
        cell.update(with: presenter.posts[indexPath.row])
        return cell
    }

    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        presenter.didSelectRowAt(indexPath: indexPath)
    }
}
```

Реалізація головних функцій делегата таблиці. Вони описують логіку відображення рядків та відповідають за обробку подій у таблиці.

Лістинг 2.16 – Головний інтерфейс комунікації модуля News

```
protocol NewsPresenter {

    var posts: [Post] { get }

    var isLoading: Bool { get }

    func didSelectRowAt(indexPath: IndexPath)

    func infoTapped()
}

//MARK: - NewsPresenter
extension NewsPresenterImp: NewsPresenter {

    var posts: [Post] {
        state.posts
    }

    var isLoading: Bool {
        state.isLoading
    }

    func didSelectRowAt(indexPath: IndexPath) {
        flow?.navigate(to: .safari(state.posts[indexPath.row].link))
    }
```

```
    func infoTapped() {
        flow?.navigate(to: .info)
    }
}
```

Інтерфейс відповідає за навігацію та відображення даних про новини. Наступна функція виконує запит до серверу для отримання новин.

Лістинг 2.17 – Функція запиту до серверу для отримання новин

```
func getPosts() {
    apiClient
        .request(PostsApi.get)
        .subscribe(
            onSuccess: { [weak self] (response: PostsResponse) in
                self?.state.posts = response.posts
                    .filter({ !$0.link.contains("/ru/")})
                    .filter({ !$0.link.contains("/en/")})
                    .map({
                        let date = $0.date.replacingOccurrences(of: "T", with: " ")

                        let dateFormatterGet = DateFormatter()
                        dateFormatterGet.dateFormat = "yyyy-MM-dd HH:mm:ss"

                        let dateFormatterPrint = DateFormatter()
                        dateFormatterPrint.dateFormat = "MM.dd.yyyy"

                        let finalDate: String

                        if let formattedDate = dateFormatterGet.date(from: date) {
                            finalDate = dateFormatterPrint.string(from: formattedDate)
                        } else {
                            finalDate = ""
                        }

                        return .init(
                            id: $0.id,
                            date: finalDate,
                            link: $0.link,
                            title: $0.title)
```

```
                })
            self?.state.isLoading = false
            self?.view?.update()
        },
        onError: { [weak self] in
            self?.state.isLoading = false
            self?.view?.update()
            print($0)
        })
        .disposed(by: disposeBag)
}
```

Після отримання відповіді від сервера функція перетворює інформацію у необхідних для відображення на екрані вид.

2.9 Платформа UIKit

Усі екрани, реалізовані за допомогою класу UIViewController, – це клас, який управляє ієрархією презентацій для додатків UIKit (https://developer.apple.com/documentation/uikit/uiviewcontroller). Усі класи для підключеного інтерфейсу надаються платформою UIKit. Вона пропонує архітектуру вікон та презентацій для реалізації інтерфейсу, інфраструктуру обробки подій для постачальників Multi-Touch та інших типових вводів у додатку, а також основний цикл виконання, необхідний для управління взаємодіями між користувачем, системою та додатком. Інші функції, що пропонує платформа, містять підтримку анімації, підтримку документів, підтримку малювання та друку, інформацію про поточне обладнання, управління текстом та відображенням.

3 ТЕСТУВАННЯ ПРОЕКТУ

Початкові умови роботи додатку не мають даних про предмети та дедлайни. Тому на головному екрані виводиться відповідне повідомлення (рис.3.1). Тестування будемо проводити на прикладі функціонала «Додавання предмету».

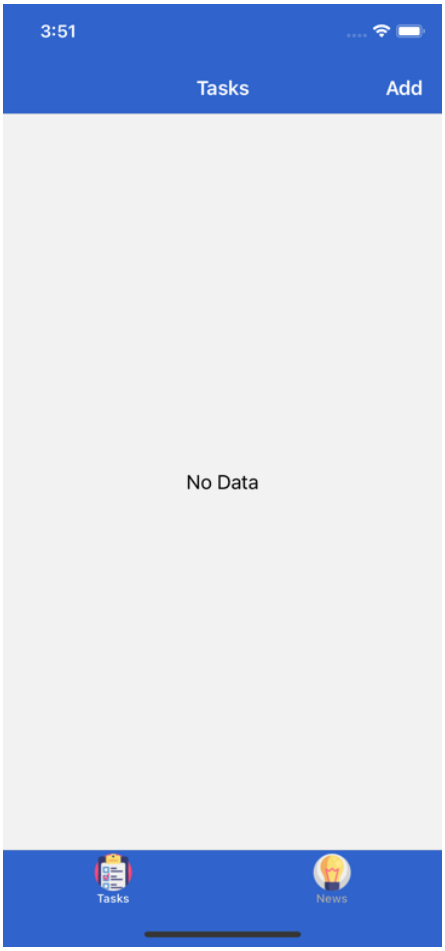

Рисунок 3.1 – Головний екран в початкових умовах

На головному екрані натискаємо кнопку “Add”. Повинно відкритися вікно “CreateNewTask”. Воно містить поля для назви предмету та кількості робіт (рис.3.2).

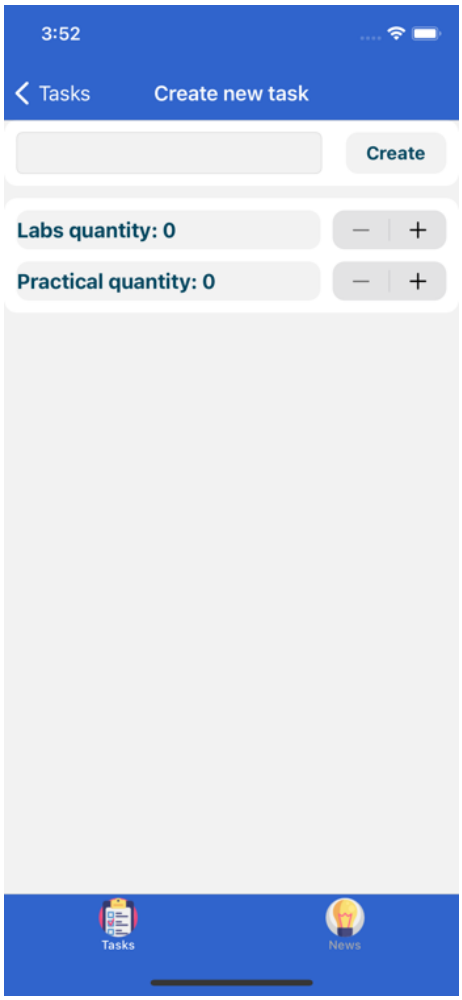

Рисунок 3.2 – Екран для додавання предмету

В поле з назвою введемо слово «Programming». Поля для кількості робіт порожні. Менше нуля робіт бути не може, тому при порожніх полях кнопка «-» є неактивною. За допомогою кнопок «+» та «-» встановимо 2 лабораторні роботи та 1 практичне завдання. На екрані повинні з’явитися три клітинки зі створеними роботами (рис.3.3). Кожна клітинка має дані про дату дедлайну. При створенні клітинок дата обирається автоматично та вказує на день створення роботи у додатку. Якщо на даному етапі натиснути на кнопку «-» біля поля з кількістю лабораторних робіт, то зі списку дедлайнів прибереться клітинка, яка була створена останньою (рис.3.4).

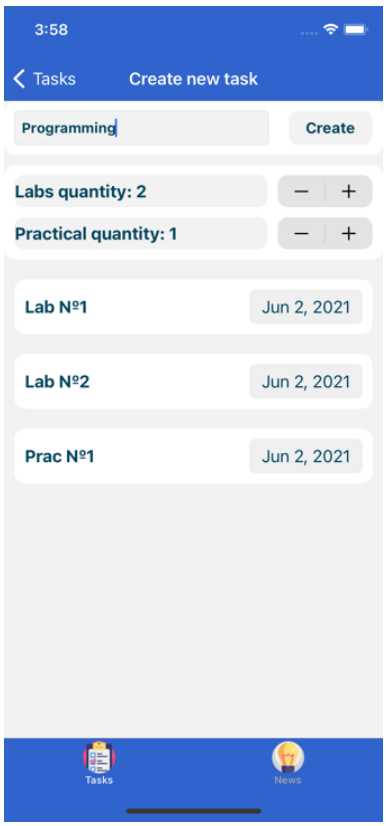

Рисунок 3.3 – Екран додавання предмету з внесеними даними

Рисунок 3.4 – Тест кнопки «-» при створенні списку робіт

Щоб змінити дату дедлайну роботи, треба натиснути на дату в клітинці. Відкриється календар (рис.3.5). Стрілочками можна обрати наступні місяці. Обрана дата виділяється блакитним колом. Встановимо дату для лабораторної роботи – 10 червня, а для практичного завдання – 15 серпня.

Рисунок 3.5 – Вибір дати дедлайну для лабораторної роботи

При закритті календарю на екрані “CreateNewTask” в переліку завдань оновлюються дані (рис.3.6).

Рисунок 3.6 – Оновленні дані про завдання

Пісня введення всіх даних про предмет, клітинка предмету створюється кнопкою «Create». Тепер новий предмет відображено на головному екрані. Пісня натиснення на кнопку «Create» дане вікно закривається. Застосунок переносить нас на головний екран з одним пунктом в переліку предметів (рис.3.7). Клітинка предмету містить назву предмету, шкалу прогресу та загальну кількість завдань.

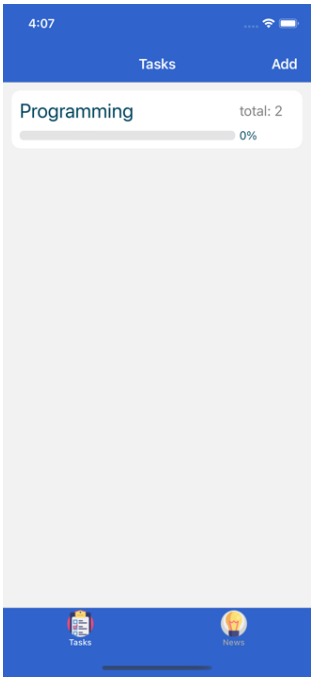

Рисунок 3.7 – Головний екран після додання предмету

При натисканні на предмет, відкриється вікно з інформацією про пункт (рис.3.8). У вікні відображено назву предмету та перелік завдань.

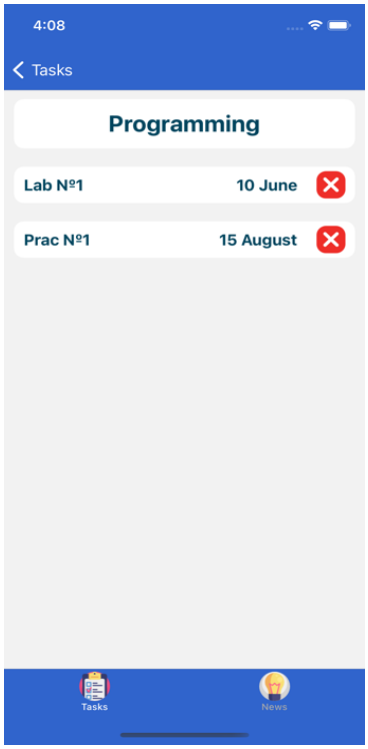

Рисунок 3.8 – Додаткова інформація про предмет

Біля кожного завдання в переліку є позначка про виконання. Автоматично встановлено, що завдання не виконано. Коли завдання чи робота виконані, користувач сам вказує на це, змінивши стан чекбоксу. Змінимо стан виконання практичного завдання (рис.3.9).

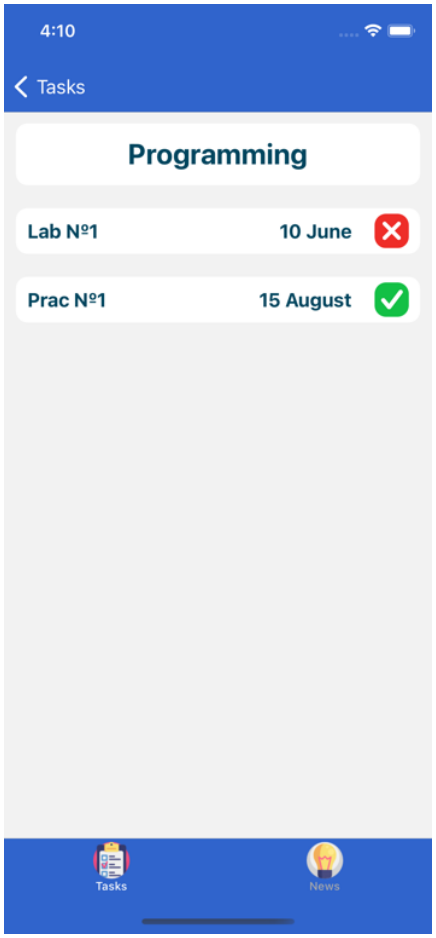

Рисунок 3.9 – Тестування роботи чек боксу

Після таких змін інформація про предмет на головному екрані оновлюється. Шкала прогресу вказує, що половина завдань с предмету успішно виконана (рис.3.10). Повернемося на головний екран за допомогою кнопки «Tasks».

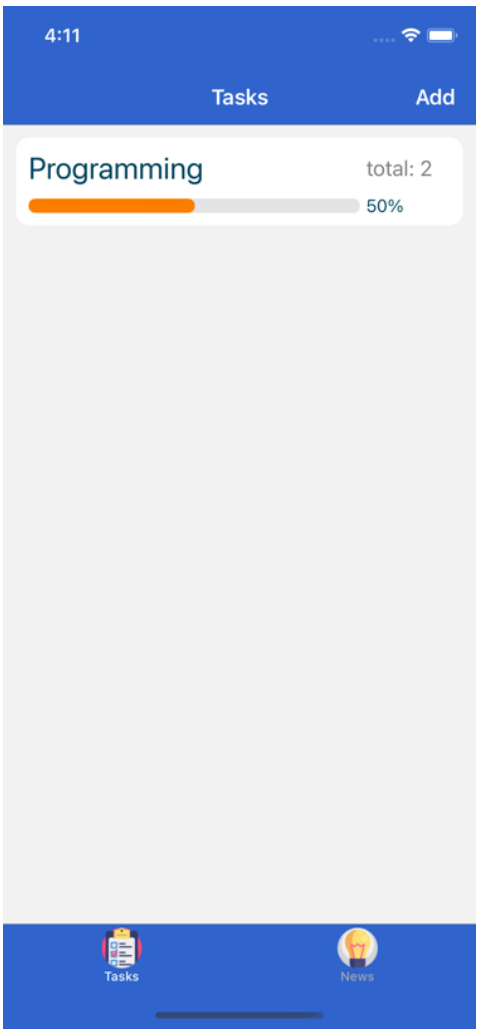

Рисунок 3.10 – Перелік предметів з частково виконаними завданнями

Для подальшого тестування було додано ще три предмети за приведеним алгоритмом.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було спроектовано модель аналізу виконання графіку навчального процесу з використанням мобільного застосунку в рамках реалізації хмарних сервісів кіберуніверситету.

Як результат спроектовано мобільний застосунок з обліком успішності навчання студента. Розглянуто обрані інструменти програмування. Детально розглянуто структуру мобільного додатку. Реалізовано чотири користувальних екрани з відповідним функціоналом. Застосунок надає можливість корегувати перелік предметів, визначати дедлайни та помічати виконанні завдання. З кожного предмета в користувача є наглядний результат навчання у вигляді шкали прогресу, яка визначається відношенням виконаних завдань до загальної їх кількості.

Розроблене програмне забезпечення дозволяє покращити самоконтроль студентів у процесі навчання. Наочне відображення прогресу з курсів та попереднє нагадування про дедлайни позитивно відобразиться на самопочутті користувача. Студент буде розуміти, скільки йому потрібно часу на виконання завдань та скільки він вже встиг виконати вчасно. Візуальне зображення успіху зменшує тиск на студентів під кінець навчального року та підчас сесії та екзаменів.

В представленій роботі надано конкретні лістинги коду програмної реалізації та проведено тестування додатку.

Серед технологічних сервісів розумного кіберуніверситету, визначених у роботі, видокремлюється хмарний кібер-сервіс оцінки якості освітніх процесів та компонентів, онлайн-перевірки знань та вмінь. У межах даного підходу запропоноване у роботі рішення може бути імплементовано як компонент хмарного сервісу, що підтверджує його практичну значущість.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хаханов В.И. Киберсоциальная система – умный киберуниверситет / В.И. Хаханов, Е.И. Литвинова, С.В. Чумаченко, А.С. Мищенко // Радіоелектронні і комп'ютерні системи. – 2016. – № 5 (79). – С.187-194.

2. Мясникова Т.С., Система дистанционного обучения MOODLE / Т.С. Мясникова, С.А. Мясников. – Харьков, 2008. – 232 с.

3. William, H. Rice IV., Moodle 2.0 E-Learning Course Development. – Packt Publishing Ltd, 2011. − 344 p.

4. Apple Developer Documentation [Електронний ресурс] / Apple Developer – Режим доступу: www / URL: https://developer.apple.com/documentation. – 01.05.2021. − Загол. з екрану.

5. UIViewController [Electronic resource]. − Access mode: https://developer.apple.com/documentation/uikit/uiviewcontroller . – 05.05.2021. − Загол. з екрану.

6. Мытников А.Н., Технологии разработки мобильных приложений / А.Н. Мытников, Е.А. Мытникова, Л.Н. Кузнецова, С.Ю. Солин // Теория и практика современной науки. – 2016. – № 4 (10). – С. 504-507.

7. Swift. Языкпрограммирования с открытым кодом. Мощь, простота и потрясающиеприложения. [Електроний ресурс] Apple.com – Режим доступа: www / URL: https://www.apple.com/ru/swift/ . – 01.05.2021. − Загол. з екрану.