МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ГНАТІЄВИЧ Остап Володимирович

Мобільна система управління процесом навчання у ВНЗ/ Mobile system for controlling the study process in University

спеціальність: 8.05010302 - Інженерія програмного запезпечення
магістерська програма - Інженерія програмного забезпечення

Магістерська робота

Виконав студент групи ІПЗм-21
О. В. Гнатієвич
_____

Науковий керівник:
к.т.н., доцент ШПІНТАЛЬ М.Я.
_____

Магістерську роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

ЗМІСТ

ВСТУП

На даний час розвиток автоматизованих систем управління навчальним процесом стає актуальним через стрімкий розвиток як і мобільних платформ так і технологій в цілому, що дозволяє значно покращити ефективність роботи різного роду діяльності. Аналіз відомих автоматизованих систем управління навчальним процесом показує, що вони функціонують з використанням елементів традиційних інформаційних технологій, які застосовуються при оптимізації організаційної структури і методів навчання. Розроблення інформаційних технологій для систем аналізу і управління навчальним процесом є актуальною задачею, оскільки широка мережа навчальних закладів різних рівнів акредитації та форм власності зумовлює необхідність незалежного контролю за дотриманням державних стандартів в галузі освіти.

Актуальність теми. Сьогодні перед розробниками для мобільних платформ відкриваються неймовірні можливості. Мобільні телефони ще ніколи не були такими популярними, а потужні смартфони тепер загальнодоступні. В стильних і багатофункціональних пристроях такі апаратні можливості, як GPS, акселерометри і сенсорні екрани поєднуються з адекватними тарифами на передачу даних по мережі, завдяки чому можна отримати доступ до привабливої платформи і мати можливість створювати для неї інноваційні мобільні додатки.

Мета і завдання дослідження. Метою роботи є розробка ефективної автоматизованої мобільної системи управління навчальним процесом у ВНЗ, яка дасть можливість приймати управлінські рішення в режимі реального часу. Для розробки

Для досягнення мети необхідно вирішити наступні задачі:

1. Проаналізувати наявні системи з управління навчальним процесом та виявити їх переваги і недоліки;
2. Розробка методу аналізу навчального процесу в режимі реального часу;
3. Розробка та обґрунтування структури мобільної автоматизованої системи управління навчальним процесом;
4. Реалізувати мобільну автоматизовану систему управління навчальним процесом;

Практичне значення одержаних результатів. Практичне значення одержаних результатів полягає в тому, що на основі запропонованих та удосконалених процедур по управлінню навчальним процесом у ВНЗ, створено програмний продукт, який застосовується для підвищення ефективності роботи працівників безпосередньо задіяних в навчальному процесі.

Об'єктом дослідження є моніторинг успішності студентів протягом навчання та ефективної роботи викладачів. Основне її призначення обробляти навчальні дані стдуентів та викладачів в режимі реального часу.

Предметом дослідження є мобільна система управління навчальним прцоесом у ВНЗ.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

1.1 Основні поняття навчального процесу та пов’язаних систем управління

Автоматизована система управління (скорочено АСУ) - комплекс апаратних і програмних засобів, а також персоналу, призначений для управління різними процесами в рамках технологічного процесу, виробництва, підприємства. АСУ застосовуються в різних галузях. Термін «автоматизована», на відміну від терміну «автоматична», підкреслює збереження за людиною-оператором деяких функцій, або найбільш загального, цілеспрямованого характеру, або не піддаються автоматизації. Найважливіше завдання АСУ - підвищення ефективності управління об'єктом на основі зростання продуктивності праці і вдосконалення методів планування процесу управління. Розрізняють автоматизовані системи управління об'єктами і функціональні автоматизовані системи, наприклад, проектування планових розрахунків, матеріально-технічного постачання тощо.

У загальному випадку, систему управління можна розглядати у вигляді сукупності взаємопов'язаних управлінських процесів і об'єктів. Узагальненої метою автоматизації управління є підвищення ефективності використання потенційних можливостей об'єкта управління. Таким чином, можна виділити ряд цілей:

З розвитком комп'ютерних-інформаційних технологій з'являється багато нових можливостей для розширення бізнесу, задля збільшення прибутків фірми та розширення клієнтської бази. Одним із способів впровадження інформаційних технологій у процес навчання вищого

навчального закладу є створення електронного журналу відвідувань студентів. Це допоможе краще слідкувати за відвідуваністю студентів та дозволить уникнути помилок через «людський фактор». Відтак, проблема трансформації системи вищої освіти України в контексті вимог Болонської декларації є надзвичайно актуальною.

Проблема обробки величезного масиву інформаційних потоків, наштовхнула авторів на думку щодо розробки та запровадження комп'ютерної програми автоматизації обліку навчальних досягнень студентів, яка дозволила б:

- спростити виконання рутинних трудомістких обчислювальних операцій, зокрема елементарних арифметичних розрахунків;
- суттєво скоротити витрати часу викладача на обслуговування необхідного навчально-облікового документообігу;
- підвищити рівень достовірності та оперативності інформації;
- забезпечити викладача своєчасною, повною та достовірною інформацією.

Якісна підготовка фахівців у ВНЗ неможлива без контролю успішності протягом кожного семестру. Ефективна система контролю поточної успішності дозволяє не тільки оцінювати виконання навчального плану кожним студентом ВНЗ, а й оцінити якість реалізованих у ВНЗ освітніх програм, своєчасно звернути увагу на труднощі студентів у освоєнні окремих навчальних дисциплін, отримати показники ефективності кожного викладача.

Звітність по поточній успішності студентів включає в себе результати виконання контрольних заходів по кожній з дисциплін навчального плану, що вивчаються студентами за освітньою програмою даної спеціальності (спеціалізації) протягом усього терміну навчання в Університеті. Види контрольних заходів можуть включати, наприклад: рубіжний контроль, проміжну поточну атестацію, семінарські заняття, лабораторний практикум,

колоквіум, виконання домашніх завдань, курсового і дипломного проектування, виконання бакалаврських та магістерських кваліфікаційних робіт і т.д.

Як правило, така звітність здійснюється протягом усього навчального семестру, а для студентів, які мають заборгованості по виконанню контрольних заходів або індивідуальних графік їх здачі, вона продовжується до моменту повної ліквідації академічної заборгованості або відрахування даного студента за академічну неуспішність.

Перелік дисциплін, види семестрової звітності по ним і терміни виконання контрольних заходів строго відповідають навчальним планом, а конкретно його семестровими «відрізках».

З першого дня навчального семестру в модулі «Поточна успішність» підсистеми «Супроводження навчального процесу» на кожну навчальну групу формується пакет електронних форм, за допомогою яких здійснюється контроль за сторонами беруть участь в навчальному процесі. В електронні форми викладачами, які реалізують навчальний процес, вносяться відомості щодо поточної успішності кожного студента групи.

В практику власної викладацької діяльності було запроваджено авторський «Електронний журнал», створений на платформі Microsoft Excel з використанням елементів мови Visual Basic for Applications. Вибір платформи зумовлено присутністю операційних систем Microsoft Windows XP/Vista та Microsoft Excel на переважній більшості комп’ютерів, високою функціональністю та простотою реалізації.

Електронний журнал – це інформаційна система, завдяки якій викладачі, студенти та їхні батьки стають набагато ближчими. Електронний журнал реалізовано як систему взаємопов’язаних електронних таблиць до яких вносяться вихідні данні: списки студентів академічної групи; теми

лекцій, практичних та лабораторних занять; всі форми поточного та підсумкового контролю; шкали оцінювання.

Протягом семестру в електронному журналі фіксується відвідування та отримані студентами бали за всі форми контролю, з врахуванням їх своєчасності. Електронний журнал забезпечує автоматизований розрахунок в діалоговому режимі кількості балів поточного і підсумкового контролю, рейтингу студентів та формування підсумкової оцінки в 100-бальній, 4-бальній й європейській системах.

Електронний журнал також дозволяє проводити докладний аналіз успішності студентів в розрізі академічних груп, навчальних модулів, лабораторних, практичних та контрольних робіт, результатів виконання тестових завдань.

Діючий електронний журнал має додатковий психологічний вплив на викладачів – вони відчувають полегшення у виконанні рутинних розрахунків, суттєву економію час та зменшення механічних помилок під час розрахунків, що особливо важливо для гуманітарних вузів. Це інструмент, який багато в чому робить процес управління освітою більш оперативним і зручним, дозволяючи швидко і одночасно інформувати про поточні зміни в навчальному закладі всіх учасників освітнього процесу. Завдяки пошуковій системі Google можна створити свій власний електронний журнал. Суть такого журналу най- краще розкриють його переваги. Переваги журналу: отримання батьками інформації про відвідування та оцінки, завдяки чому вони можуть легко контролювати успішність своєї дитини; наявність доступу до журналу в будь-який час та в будь-якому місці, де є Інтернет; для викладача стає легко порахувати підсумкові оцінки всіх дітей витративши на це менше ніж хвилину свого часу; викладач може створювати коментарі до журналу; для дітей та їхніх батьків не буде «сюрпризом» їх підсумкова оцінка; в будь-який зручний для викладача час та місці вносити дані;

відкритість в оцінюванні. У своїй професійній діяльності я користуюсь електронним журналом. Це полегшує моніторинг навчальної діяльності, а також допомагає при підготовці необхідної документації при здачі звітів в кінці навчального року тощо.

1.2 Аналіз відомих програмних засобів автоматизації управління навчальним процесом

На сьогоднішній день програмних засобів, що так чи інакше автоматизують процес управління навчальним процесом є немало. Однозначних лідерів серед них немає тому для аналізу було обрано наступні системи: АСУ «МКР», "Електронний журнал КНЕУ", "Портал АСУ ім. Ярослава Мудрого". Короткий аналіз кожного з них для представлення їх можливостей в вирішення необхідних нам задач. Усі перелічені системи розроблені з метою спрощення процесу управління навчанням, як для студентів так і для викладачів.

Розглянемо детальніше систему «Автоматизована система управлання навчальним процеосм "МКР"». Загальний вигляд головної сторінки, якої зображено на рисунку 1.1.

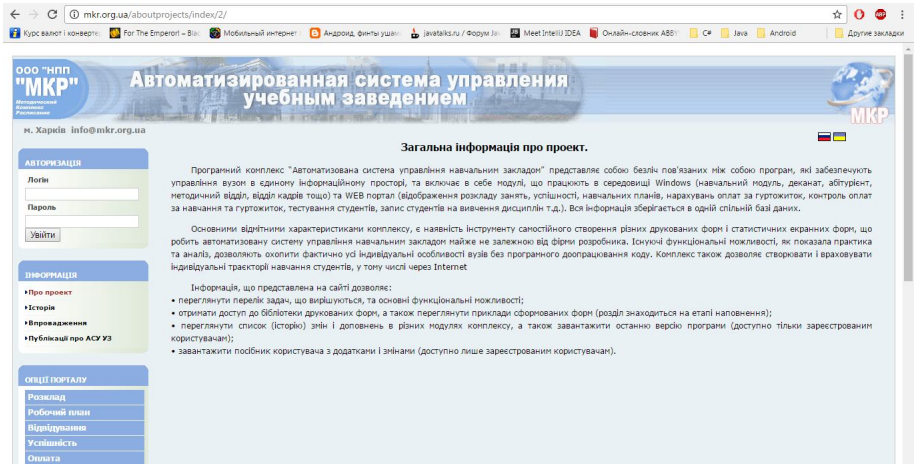


Рисунок 1.1 – Загальний вигляд головної сторінки Автоматизована система управлання навчальним процеосм "МКР"

На головній сторінці бачимо перелік новин та різні навігаційні блоки, які дозволяють користувачу здійснювати певні дії:

- Авторизація;
- Інформація;
- Опції порталу;

Обираючи блок «інформація» може бути завантажена одна із наступних веб-сторінок: «Про проект», «Історія», «Впровадження», «Публікації про АСУ УЗ». На ній представлено інформацію про те де дана система уже використовується.

Обираючи блок «опції порталу» може бути завантажена одна із наступних веб-сторінок: «Розклад», «Успішність», «Відвідування», «Зареєструватися», «Список студентів», «Moodle», та інші.

Для реєстрації студента необхідно ввести свій ідентифікаційний номер, що видається деканатом (рисунок 1.2).

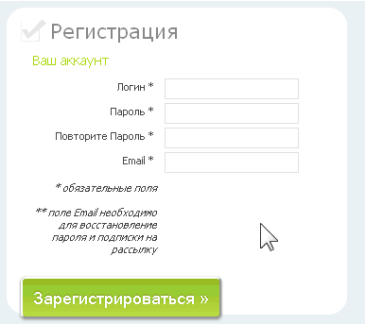


Рисунок 1.2 – Вікно реєстрації

Після заповнення форми та при умові ведення коректних даних, що не суперечать полям заповнення виведеться повідомлення про успішну реєстрацію та додаткову інформацію про дії у раз ситуації втрати пароля (рисунок 1.3).

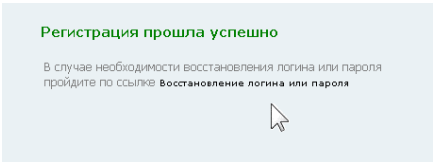


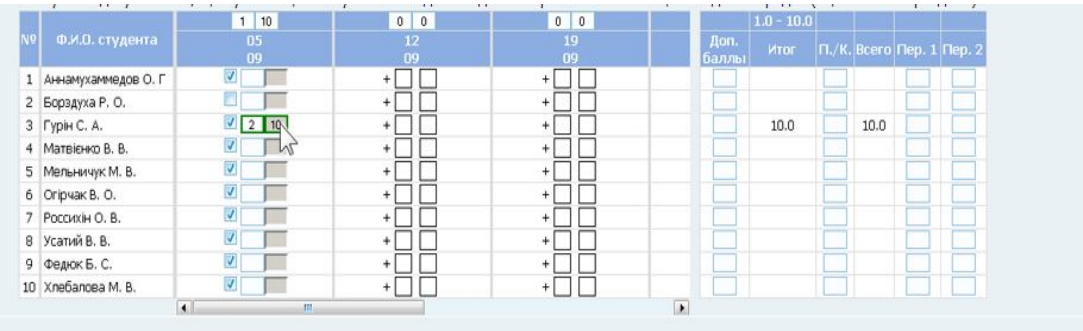
Рисунок – 1.3 – Успішне заверення реєстрації

Також можна переглянути електронний журнал успішності студентів. Дане головне вікно поділене на блоки:

- Блок 1 – призначений для вибору навчального року та навчального семестру для перегляду журналу успішності;
- Блок 2 - призначений для вибору дисципліни і групи, що вивчає обрану дисципліну;
- Блок 3 - призначений для введення успішності студентів групи. Клітинки, які відповідають заняттям поточної дати відображаються жовтим кольором.

Електронний журнал може мати наступний вигляд. Для проставлення відвідуваності і оцінок необхідно в колонці з датою вказати мінімальний і максимальний можливий бал за заняття, після лівої кнопкою миші натиснути

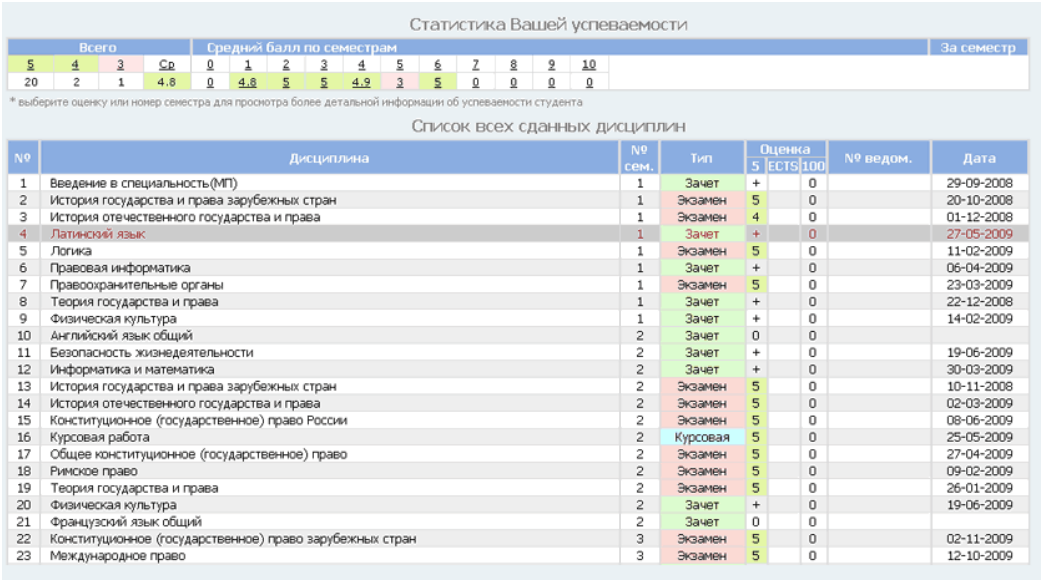
на дату заняття, щоб у колонці активувати поля для введення. При виставленні оцінок є два осередки (оцінка та її перездача) (рисунок 1.4).



№	Ф.И.О. студента	1 10 05 09	0 0 12 09	0 0 19 09		1.0 - 10.0 Доп. баллы	Итог	П./К.	Всего	Пер. 1	Пер. 2
1	Аннамухаммедов О. Г		+	+							
2	Борздуха Р. О.		+	+							
3	Гурін С. А.	2 10	+	+			10.0		10.0		
4	Матвієнко В. В.		+	+							
5	Мельничук М. В.		+	+							
6	Огірчак В. О.		+	+							
7	Россихін О. В.		+	+							
8	Усатий В. В.		+	+							
9	Федюк Б. С.		+	+							
10	Хлебалова М. В.		+	+							

Рисунок 1.4 – Нижній блок журналу успішності

Вікно успішності представлено на рисунку 1.5.



Статистика Вашей успеваемости

Всего				Средний балл по семестрам											За семестр
5	4	3	Ср	0	1	2	3	4	5	6	7	8	9	10	
20	2	1	4.8	0	4.8	5	5	4.9	3	5	0	0	0	0	

* выберите оценку или номер семестра для просмотра более детальной информации об успеваемости студента

Список всех сданных дисциплин

№	Дисциплина	№ сем.	Тип	Оценка 5	ECTS	100	№ ведом.	Дата
1	Введение в специальность(МП)	1	Зачет	+		0		29-09-2008
2	История государства и права зарубежных стран	1	Экзамен	5		0		20-10-2008
3	История отечественного государства и права	1	Экзамен	4		0		01-12-2008
4	Латинский язык	1	Зачет	+		0		27-05-2009
5	Логика	1	Экзамен	5		0		11-02-2009
6	Правовая информатика	1	Зачет	+		0		06-04-2009
7	Правоохранительные органы	1	Экзамен	5		0		23-03-2009
8	Теория государства и права	1	Зачет	+		0		22-12-2008
9	Физическая культура	1	Зачет	+		0		14-02-2009
10	Английский язык общий	2	Зачет	0		0		
11	Безопасность жизнедеятельности	2	Зачет	+		0		19-06-2009
12	Информатика и математика	2	Зачет	+		0		30-03-2009
13	История государства и права зарубежных стран	2	Экзамен	5		0		10-11-2008
14	История отечественного государства и права	2	Экзамен	5		0		02-03-2009
15	Конституционное (государственное) право России	2	Экзамен	5		0		08-06-2009
16	Курсовая работа	2	Курсовая	5		0		25-05-2009
17	Общее конституционное (государственное) право	2	Экзамен	5		0		27-04-2009
18	Римское право	2	Экзамен	5		0		09-02-2009
19	Теория государства и права	2	Экзамен	5		0		26-01-2009
20	Физическая культура	2	Зачет	+		0		19-06-2009
21	Французский язык общий	2	Зачет	0		0		
22	Конституционное (государственное) право зарубежных стран	3	Экзамен	5		0		02-11-2009
23	Международное право	3	Экзамен	5		0		12-10-2009

Рисунок 1.5 – Звіт успішності журналу

Також є можливість переглянути загальну статистику відвідування студента будь-якої групи у зазначеному семестрі або у зазначенному місяці. Тепер розглянемо детальніше систему «АСУ КНЕУ». Загальний вигляд головної сторінки, якої зображений на рисунку 1.6.

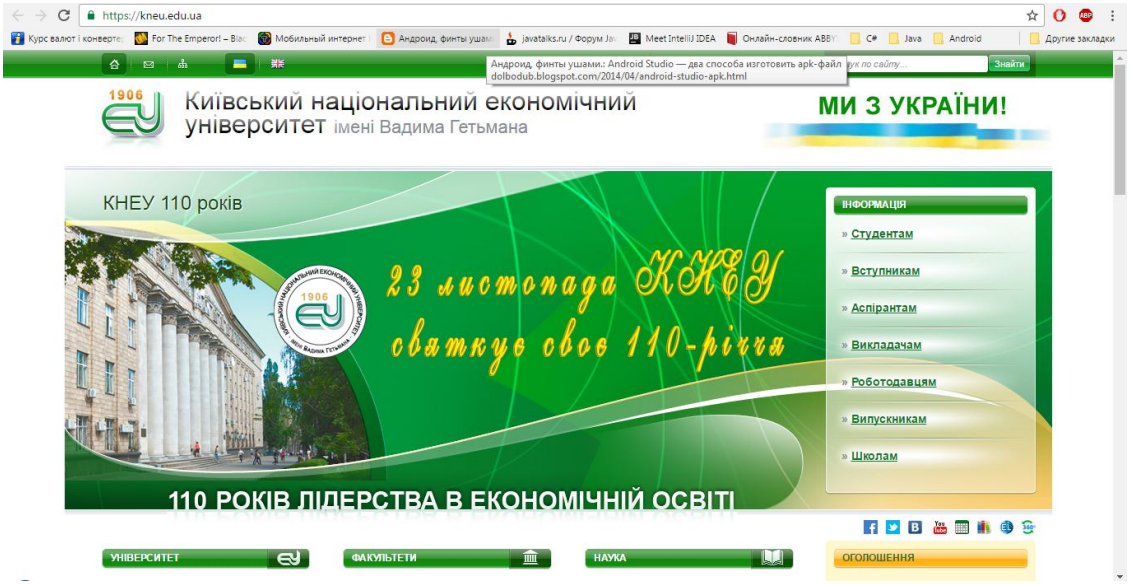


Рисунок 1.6 - Загальний вигляд головної сторінки системи «АСУ КНЕУ»

На головній сторінці бачимо перелік новин та різні навігаційні кнопки, які дозволяють користувачу переходити на різні тематичні розділи веб-сайту. Обравши пункт «Студентам» відкриється веб-сторінка на якій буде відображена різного роду інформація, як наприклад: новини щодо процесу навчання у ВНЗ, також посилання на наступну важливу інфморацію:

- Журнал успішності студентів(2-х видів);
- Навчально-методичні матеріали для бакалаврів;
- Зразки індивідуальних планів для різних форм навчання студентів;
- Та інша не менш корисна інформація по навчанню студентів даного ВУЗу.

Після переходу на веб-сторінку "Журнал успішності", завантажується форма подачі заявки (рисунок 20). Для входу в Електронний журнал необхідно зареєструватися на сайті КНЕУ. Реєстрація дозволить отримати доступ до ресурсів сайту, електронного журналу, бібліотеки. Якщо студент ще не зареєстрований то відкритті посилання «Ще не зареєстровані?» відкривається вікно вибору одного із 3-ох типів користувачів системи (рисунок 1.7)

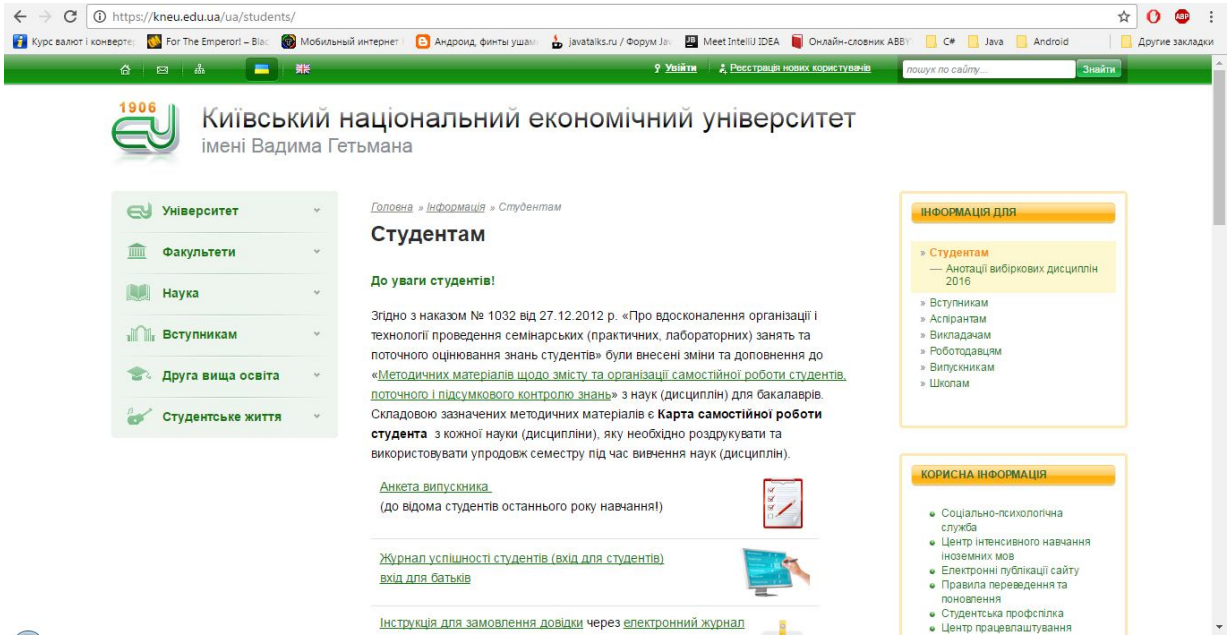


Рисунок 1.7 – Пункт «студенти» КНЕУ

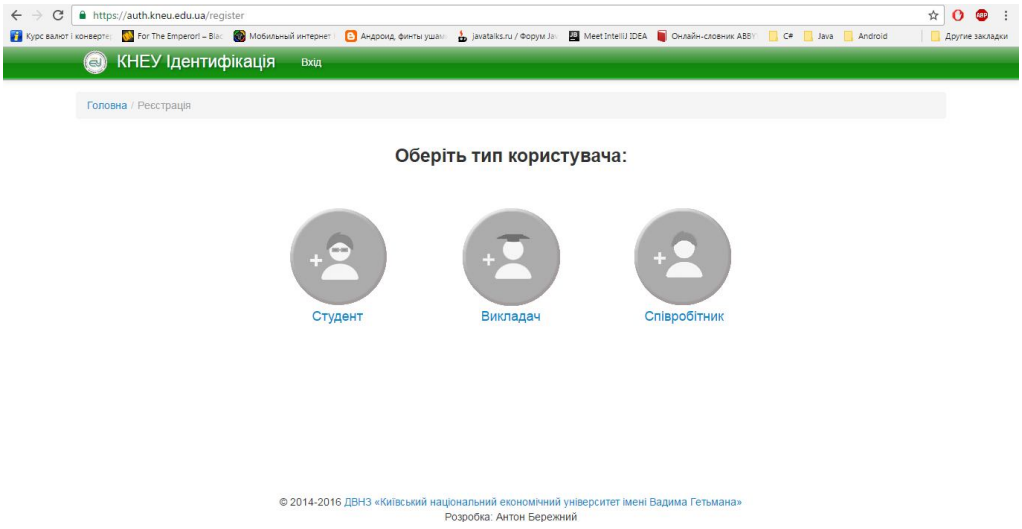


Рисунок 1.8 – Реєстрація певного типу користувача

Після обрання типу користувача системи відкривається наступне вікно системи в якому необхідно ввести певні дані (рисунок 1.9.

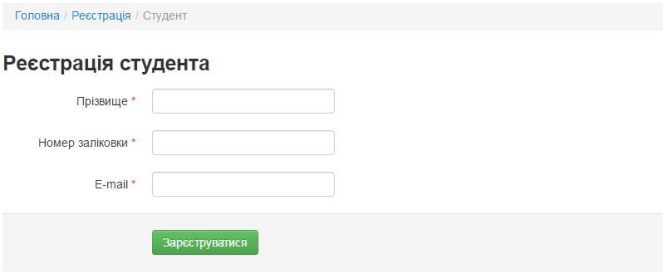


Рисунок 1.9 – Реєстраційне вікно системи

Після виконання даних дій можна вернутися на головну сторінку порталу. Також однією з наступних дій після успішної реєстрації перегляд успішності студента та його відвідуваність.

Основне вікно мобільного клієнта даної системи представлено на рисунку. З якого можна зрозуміти, що основне призачення даного додатку це перегляд оцінок без отримання змоги внести внести оцінки, тобто лише для студента(рисунок 1.10).

Рисунок 1.10 – Вікно ел.журналу КНЕУ мобільний додаток

Далі, розглянемо програмну систему «Портал АСУ навчальним процесом Національного юридичного університету імені Ярослава Мудрого». Загальний вигляд головної сторінки цієї системи показано на рисунку 1.11.

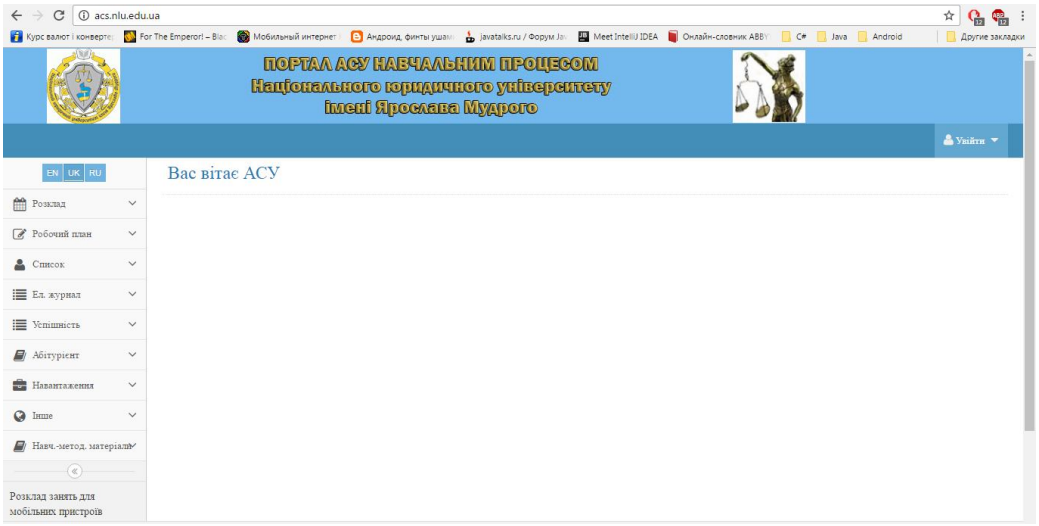


Рисунок 1.11- Загальний вигляд системи «Портал АСУ навчальним процесом Національного юридичного університету імені Ярослава Мудрого»

Вигляд форми для подачі заявки на реєстрацію студента чи викладача показано на рисунку 1.12.

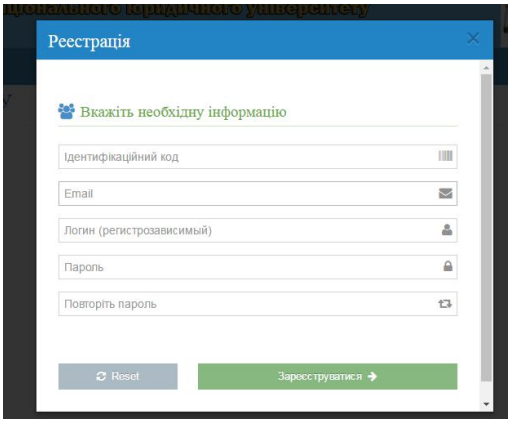


Рисунок 1.12 - Форма подачі заявки «АСУ імені Ярослава Мудрого»

Перейшовши на сторінку "Ел. Журнал" бачимо форму заповнення на отримання даних по статистиці відвідуваності студентів певної категорії, необхідно заповнити наступні дані.

У таблиці 1.1 наведено порівняльну характеристику розглянутих програмних продуктів (аналогів).

Таблиця 1.1

Порівняльна характеристика програмних продуктів

Назва програмного продукту	АСУ імені Ярослава Мудрого	Електронний журнал КНЕУ	Автоматизована система управлання навчальним процеосм "МКР"
Версії продукту	Невказано	Не вказано	Не вказано
Інтерфейс користувача	Web-система	Web-система	Web-система
Вибір типу користувача	Відсутній вибір	Наявний вибір	Відсутній вибір
Мобільна версія системи	-	+	-

Як видно з опису систем, у ній застосовуються не тільки технології побудови веб-ресурсних додатків , але також технології розробки серверних доадктів. З даної таблички можна зробити висновок, що систем «Електронний журнал КНЕУ» має доволі хороший функціонал і з поміж 3-ох систем має вибір типу користувача при реєстрації», з іншого боку «АСУ імені Ярослава Мудрого» має більий функціонал проте відсутній вибіор типу користувача при реєстрації. Проте тільки «Електронний журнал КНЕУ» має мобільну версію системи.

1.3 Аналіз мобільних пристроїв для виконання мобільної системи управління навчальним процесом

Сьогодні існує достатньо велика різноманітність операційних систем, однак в комунікаторах і смартфонах широкого розповсюдження отримали.
Проте світ мобільних платформ сильно фрагментований, тут виділяються дві основні операційні системи - Android і iOS, а також платформу Windows Phone / Windows 10 Mobile. Є певні статистичні дані, що значна частина мобільних додатків створюється більш ніж для однієї платформи, наприклад, для Android і iOS. Однак неминуче розробники стикаються з наступними труднощами:
- відмінність в підходах побудова графічного інтерфейсу так чи інакше впливає на розробку. Розробники змушені підлаштовувати додаток під вимоги до інтерфейсу на конкретній платформі;
- різні API - розбіжність у програмних інтерфейсів і реалізації тих чи інших функциональностей також вимагає від програміста облік цих специфічних особливостей різних платформи для розробки. Наприклад, щоб створювати додатки для iOS нам необхідна відповідне середовище - Mac OS X і ряд спеціальних інструментів, типу XCode. А в якості мови програмування вибирається Objective-C або Swift. Для Androida ми можемо використовувати найрізноманітніший набір середовищ - Android Studio, Eclipse і т.д. Але тут для переважної більшості додатків застосовується Java. Найбільш популярні ОС для мобільних пристроїв – це Android, Apple iOS, Windows Phone.
Компанія Apple розробила для своїх пристроїв власну операційну систему iOS, - це фірмова розробка, які встановлюють тільки на пристрої Apple. Інтерфейс системи заснований на прямому маніпулюванні, тільки в сенсорних пристроях. Переваги операційної системи iOS:

- App Store – маркет додатків, де існують як платні так і безплатні додатки;
- оновлення відбуваються для всіх пристроїв з ОС iOS;
- система розрахована на те, щоб користувач просто користувався, а не заглиблювався в тонкощі роботи системи. Тому всі версії iOS проходять серйозне тестування;
- всі пристрої на iOS, як і аналогічні пристрої від Apple, мають дуже низьку енергозатрати;
- операційна система iOS має високу швидкодію, вона працює стабільно і не схильна до вірусних атак.

До недоліків пристрою на iOS варто віднести неможливість переносу на нього даних без встановленої на ньому програмі iTunes. Жоден мобільний пристрій від Apple не підтримує прямого копіювання файлів в пам'ять, також в смартфонах на iOS відсутні слоти для додаткових карт пам'яті. Великим недоліком є те, що iOS не підтримує Adobe Flash Player і Java. Вартість такого пристрою завжди вища ніж в конкурентів.

Компанія Microsoft не перший рік на ринку мобільних пристроїв, але попередні версії Windows Mobile не увінчались великим успіхом. Тому зовсім недавно вийшла ОС для смартфонів Windows Phone, якій пророкують велике майбутнє.

Переваги ОС Windows Phone:

- зручна і проста синхронізація з сервісами Microsoft. В комплект поставки смартфонів на цій ОС входять додатки Windows Live, Outlook, Exchange, Office і т.д.;
- ОС Windows Phone на мобільних пристроях показують дуже високу

У список мінусів мобільних пристроїв на ОС Windows Phone можна з впевненістю внести той факт, що в онлайн-магазині Marketplace на

сьогоднішній день представлена мала кількість додатків, особливо в порівнянні з асортиментом магазинів додатків Google Play і AppStore.

Android, мобільна ОС на основі ядра Linux, стала лідером продажу у 2013 р. Система була розроблена однойменною компанією, яку придбала компанія Google. Переваги пристроїв на платформі Android:

- широкий вибір представлених на ринку різноманітних пристроїв, які функціонують під даною ОС;
- дуже просто і інтуїтивно зрозуміла синхронізація зі всіма сервісами від Google. Більшість звичних користувачу веб-додатків вже є в вихідних кодах Android. Це Gmail, Google Maps, Google+, Youtube і багато інших;
- онлайн магазин Google Play пропонує для завантаження сотні тисяч утиліт і додатків, число яких постійно зростає;
- пристрої на Android дозволяють завантажувати і встановлювати додатки, які були отримані не з Google Play;
- пристрій можна підключити до комп'ютера через USB порт. Комп'ютер розпізнає пристрій як накопичувач;
- користувач має можливість оформити інтерфейс смартфона так, як йому зручно. Система дозволяє по-своєму налаштувати кожний елемент робочого столу;
- підтримка Java та Adobe Flash Player;
- відносно не висока доступна ціна пристрою на даній ОС.

До мінусів цієї ОС можна віднести її доволі велике енергоспоживання. Крім того, всі виробники намагаються по-своєму налаштувати інтерфейс пристроїв. В результаті, придбавши новинки пристроїв на Android, користувачу знадобиться витратити деякий час на освоєння «фірмового» інтерфейсу.

1.4 Аналіз архітектур розробки програмних засобів

Архітектура програмного забезпечення - це структура програми або обчислювальної системи, яка включає програмні компоненти, видимі зовні властивості цих компонентів, а також відносини між ними.

Архітектуру програмного забезпечення можна розглядати як зіставлення між метою компонента ПЗ і відомостями про реалізацію в коді. Правильне розуміння архітектури забезпечить оптимальний баланс вимог і результатів. Програмне забезпечення з добре продуманою архітектурою виконуватиме зазначені завдання з параметрами вихідних вимог, одночасно забезпечуючи максимально високу продуктивність, безпека, надійність і багато інших чинників. Документування архітектури ПО спрощує процес комунікації між зацікавленими особами, дозволяє зафіксувати прийняті на ранніх етапах проектування рішення про високорівневої дизайні системи і дозволяє використовувати компоненти цього дизайну і шаблони повторно в інших проектах.

Розглянемо наступні загальні архітектури програмного забезпечення:

- Клієнт-серверні системи;
- Web - сервіси та Web-додатки;
- Інтегровані розподілені рішення (solutions);
- Декстопне програмне зебезпечення.

В цілому для сучасного ПО характерна тенденція до значного ускладнення архітектури.

Клієнт-серверні системи набули широкого поширення вже протягом десятків років. Відомі такі основні різновиди серверів: сервер додатків, Web-сервер, сервер баз даних, сервер електронної пошти, файл-сервер і інші. Клієнт-серверна архітектура – тип архітектури програмної системи, в якій

обробка даних ділиться на дві на сторони: клієнта і сервера. Клієнт в ній використовується для відправки запитів на сервер та отримання даних від нього. Це означає, що клієнт встановлює зв'язок з сервером, за допомогою HTTP протоколу, та обмінюється з ним інформаційними пакетами. Відображення результату залежить від рангу того, хто зайшов у систему, для викладача, наприклад будуть доступні усі функції, в той же час як для студента тільки перегляд.

Інтернет - додатки призначені для виконання в Мережі. В сучасних умовах більшість з них розробляється на платформі .NET або Java, хоча деякі програмісти і фірми досі пишуть Інтернет-додатки на C. У сучасному Web-програмуванні також широко використовуються мови з динамічними типами - JavaScript, Python, Ruby, для яких характерно динамічна зміна і конструювання типів під час виконання програми, що зручно, так як відображає динамічну природу Web-додатків і Web-сайтів.

Інтегровані рішення є розподіленими програмними системами для управління інформацією і функціонуванням підприємств, фірм, банків, університетів. Для інтегрованих рішень характерна наявність модулів аутентифікації і авторизації користувачів, роботи з базами даних, роботи з мережею, реалізації бізнес-логіки. Інтегровані рішення можуть бути розроблені з використанням різних мов програмування.

Клієнт-серверна архітектура – тип архітектури програмної системи, в якій обробка даних ділиться на дві на сторони: клієнта і сервера. Клієнт в ній використовується для відправки запитів на сервер та отримання даних від нього. Це означає, що клієнт встановлює зв'язок з сервером, за допомогою HTTP протоколу, та обмінюється з ним інформаційними пакетами. Відображення результату залежить від рангу того, хто зайшов у систему, для викладача, наприклад будуть доступні усі функції, в той же час як для студента тільки перегляд.

Десктопні програми - це програми, логіка роботи яких вимагає наявності оператора (людини що працює з програмою), що містять в собі всю повну функціональність і здатні працювати окремо на будь-якій машині ізольовано від інших додатків. Microsoft Word, Excel, Блокнот, однопользовательские гри - все це приклади десктопних додатків. Для їх роботи необхідні лише достатні апаратні ресурси комп'ютера, сам додаток і набір бібліотек, що містять функції для роботи з додатком.

Десктопні програми можуть бути також і на багато користувачів. Наприклад, редактор файлів який в залежності від логіна і пароля, введених при запуску, буде давати доступ до різних файлів. І програма і файли знаходяться на одному комп'ютері, просто виробляється локальне розмежування доступу для різних користувачів.

Також потрібно розглянути архітектуру самих програмних засобів. Почнемо з першого головного - Model-View-Controller. MVC - це фундаментальний патерн, який знайшов застосування в багатьох технологіях, дав розвиток нових технологій і кожен день полегшує життя розробникам.

Вперше патерн MVC з'явився в мові SmallTalk. Розробники повинні були придумати архітектурне рішення, яке дозволяло б відокремити графічний інтерфейс від бізнес логіки, а бізнес логіку від даних. Таким чином, в класичному варіанті, MVC складається з трьох частин, які і дали йому назву. Розглянемо їх: Модель(під Моделлю, зазвичай розуміється частина містить в собі функціональну бізнес-логіку програми. Модель повинна бути повністю незалежна від інших частин продукту. Модельний шар нічого не повинен знати про елементи дизайну, і яким чином він буде відображатися), Вигляд(View)(В обов'язки Уявлення входить відображення даних отриманих від Моделі. Однак, уявлення не може безпосередньо впливати на модель. Можна говорити, що уявлення отримує доступ «тільки

на читання» до даних). Представниками звісно з певними відмінностями між собою є: MVP, MVVM та MVP.

Model-View-Presenter. Даний підхід дозволяє створювати абстракцію уявлення. Для цього необхідно виділити інтерфейс подання з певним набором властивостей і методів. Презентер, в свою чергу, отримує посилання на реалізацію інтерфейсу, підписується на події вистави і за запитом змінює модель. Приклад використання: Windows Forms.

Model-View-View Model. Даний підхід дозволяє пов'язувати елементи уявлення з властивостями і подіями View-моделі. Можна стверджувати, що кожен шар цього патерну не знає про існування іншого шару. Ознаки View-моделі:

- Двостороння комунікація з поданням;
- View-модель - це абстракція уявлення. Зазвичай означає, що властивості уявлення збігаються з властивостями View-моделі / моделі
- View-модель не має посилання на інтерфейс подання (IView). Зміна стану View-моделі автоматично змінює уявлення і навпаки, оскільки використовується механізм скріплення даних (Bindings). Приклад використання: WPF.

1.5 Постановка завдання на програму мобільна систем управління навчальним процесом

На сьогоднішній день студенти та викладачі ТНЕУ не мають можливості переглядати детальну інформацію про свій навчальним процес у університеті. Доступна лиш модульна система яка дозволяє переглянути загальні бали по основних моментах предметів, таких як модулі чи загальна

оцінка за предмет на сайті http://mod.tanet.edu.te.ua. Для перегляду потрібно мати власний обліковий запис. Після авторизації будуть відображені всі поточні оцінки студента за семестр.

Оскільки сучасні мобільні пристрої підтримують читання html-сторінок, а також доступність мобільного інтернету, дають можливість власникам мобільних пристроїв переглядати веб-сторінки, в будь-якому місці і в будь-який час.

До того ж, такі додатки можуть бути наділені додатковим функціоналом. Розробляючи будь-яке програмне забезпечення, потрібно завжди орієнтуватись на користувача.

Отже, розроблюваний продукт повинен забезпечувати виконання наступних функцій:

- використання бази даних для збереження данних системи;
- можливість перегляду персональної університетської інформації;
- підтримка реєстрації декілької типів користувачів;
- можливість перегляду списку предметів, до яких так чи інакше прикріплений користувач;
- можливість перегляду оцінки;
- можливість збереження користувачем власного розкладу;
- можливість одного з типів користувачів використнання активних опцій системи, таких як: редагування, додавання та видалення записів у базі даних.

ВИСНОВКИ ДО І РОЗДІЛУ

У першому розділі проведено аналіз основних моментів розробки автоматизованої системи управління навчальним процесом, а саме:

- описано основні моменти та поняття області автоматизованих систем управління навчальним процесом. Виявлено, як саме дані системи допомагають покращити роботу розіного роду підприємств та установ, зокрем освітніх, за рахунок зменшення можливості втрати даних, пришвидшення обробки інформації, пришвидшення роботи операторів та можливості швидкого доступу до інфморації;
- розглянуто можливі архітектурні рішення, що використовуються в даних системах, виявлено їх основні моменти, переваги та недоліки і здійснено вибір відповідно до вимог розроблюваної системи. В рамках даного проекту було обрано для загальної ідеї архітетуру клієнт-сервер, що дозволяє спростити розробку системи та відділити основне джерело данних від програми. Для самих програм було обрано архітектурне рішення, назване як «MVVM-паттерн»,що дозволяє відділити бізнес-логіку програми від її вигляду, краще розробити інтерфейс програми та підвищує пітрмику програми;
- розглянути різні мобільні платформи, що популярні зараз на ринку мобільних пристроїв, коротко описано їх основні дані, представлено основні їх момента, переваги та недоліки кожного з них. В результаті було обрано мобільну систему Android, як основну мобільну платформу для клієнта розроблюваної системи та iOS ОС, як альтернативну;
- проаналізовано основні аналоги, що можуть складати конкуренію розроблюваній система, представлено їх основні моменти, можливості, переваги та недоліки і на основі отриманої інформації, було сформовано вимоги до розроблюваного програмного продукту.

2 РОЗРОБКА АРХІТЕКТУРИ МОБІЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ

2.1 Розробка загальної структури системи

«Мобільна система управління навчальним процесом у ВНЗ» - це є клієнт-серверна програма для перегляну та керування електронного журналу навчальними данними, такими як: оцінка по предмету, присутніть чи відсутність студента на занятті, перегляд персональної інформації.

Для того, щоб користувач міг користуватися системою йому необхідно зареєструватися чи бути зареєстрованим. Для реєстрації користувача потрібно пройти наступний крок:

1. Створення облікового запис адміністратором. Ввести логін, пароль, прізвище, ім`я, ім`я по-батькові, дату народження та роль.

Після входу у систему, використовуючи персональний обліковий запис, якщо це студент то йому доступні наступні можливості: перегляд данних, як особистих так і навчальних(Прізвище ім`я, групу); список предметів , які викладаються студенту; розклад навчальних занять по курсуж; журнальний список, в якому буде відображатсия інформація про успішність студента у вигляді: дата проведення заняття, найменування заняття, відмітка за заняття, оцінка або пропуск та власне хто веде даний предмет. Також можна отримати інформацію про останнє оновлення бази даних.

В ході проектування системи програмістом створюється документація по проекту, що може включати в себе: текстовий опис, різного роду діаграми, моделі розроблюваної програми та інше. Для розробки загальної структури програми використовується мова UML та пов`зані з неї програми.

UML – це графічна мова для візуалізації, опису параметрів, конструювання і документування різноманітних систем. Діаграми створюються за допомогою

спеціальних CASE засобів. За допомогою технології UML будується єдина інформаційна модель. Серед основних типів діаграм для візуалізації моделі присутні наступні: діаграма варіантів використання (use case diagram), діаграма класів (class diagram), діаграма станів (statechart diagram), діаграма послідовностей (sequence diagram), діаграма компонентів (component diagram).

Для того, щоб максимально ясно зрозуміти як повинна працювати система, доволі часто використовують опис функціональності системи за допомогою варіантів використання (use case або ще називають прецеденти). Варіанти використання – це опис послідовності дій, які може виконувати система у відповідь на зовнішні дії користувачів чи інших програмних систем. Варіанти використання представляють функціональність системи з погляду отримання результату для користувача, тому вони дають отримати точніші результата, за рахунок чого виходить краще вистроювати функції за важливістю одержуваного результату.

Варіанти використання призначені в першу чергу для визначення функціональних вимог до системи і внаслідок управляють всім процесом розробки. Всі основні види процесів діяльності розробки таких як: аналіз, проектування та тестування виконуються на основі варіантів використання. Діаграма варіантів використання складається із акторів, для яких система виконує дії і власне дії use case, яке описує те, що актор хоче отримати від системи. Визначення акторів. Для знаходження акторів слід шукати відповіді на наступні питання:

- Хто є безпосередніми користувачами системи? Необхідно при відповіді на дане питання вказувати ролі, а не конкретних людей, які виконують ці ролі.
- З яким зовнішнім устаткуванням або програмами здійснює взаємодію система?

• Чи виконує система роботи, що активізуються настанням конкретного часу або закінченням певних періодів часу?

Провівши аналіз системи, визначили, що у розроблюваній системі буде тільки 3 актори – адміністратор, який може виконати всі функції (use case), передбачені технічним завданням; викладач, який може виконувати тіж функції, що і адміністратор окрім додавання та виделення користувачів та обмеженого редагування; студент, який може лиш переглядати введені викладачем чи адміністратором дані, якщо вони стосуються особисто його.

Між елементами діаграми варіантів використання можуть існувати різні відносини, які описують взаємодію одних акторів і варіантів використання з іншими акторами та варіантами використання. В цьому випадку цей актор звертається до кількох сервісів даної системи. У свою чергу один варіант використання може взаємодіяти з декількома акторами, надаючи для всіх них свій сервіс.

У той же час два варіанти використання, визначені в рамках однієї системи, що моделюється, також можуть взаємодіяти один з одним, однак характер цієї взаємодії буде відрізнятися від взаємодії з акторами.

Далі проведемо аналіз системи та визначимо хто і як, з нею взаємодіє. На рисунку 2.1 представлено початковий ВВ.

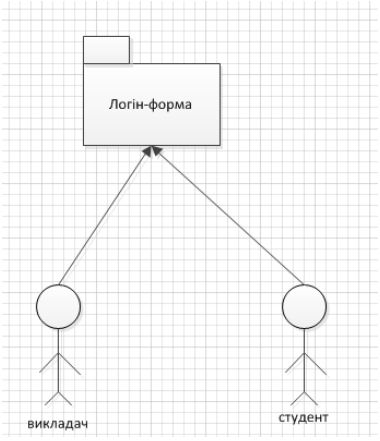


Рисунок 2.1 – Початковий варіант використання системи

Визначимо основні варіанти використання для студента та викладача. Можливі функції для студента представлені на рисунку 2.2.

Проаналізувавши усі варіанти використання до web-орієнтованої інформаційної системи для автоматизації роботи викладача у ВНЗ було сформовано наступні діаграми варіантів використання(рисунки 2.3-2.5).

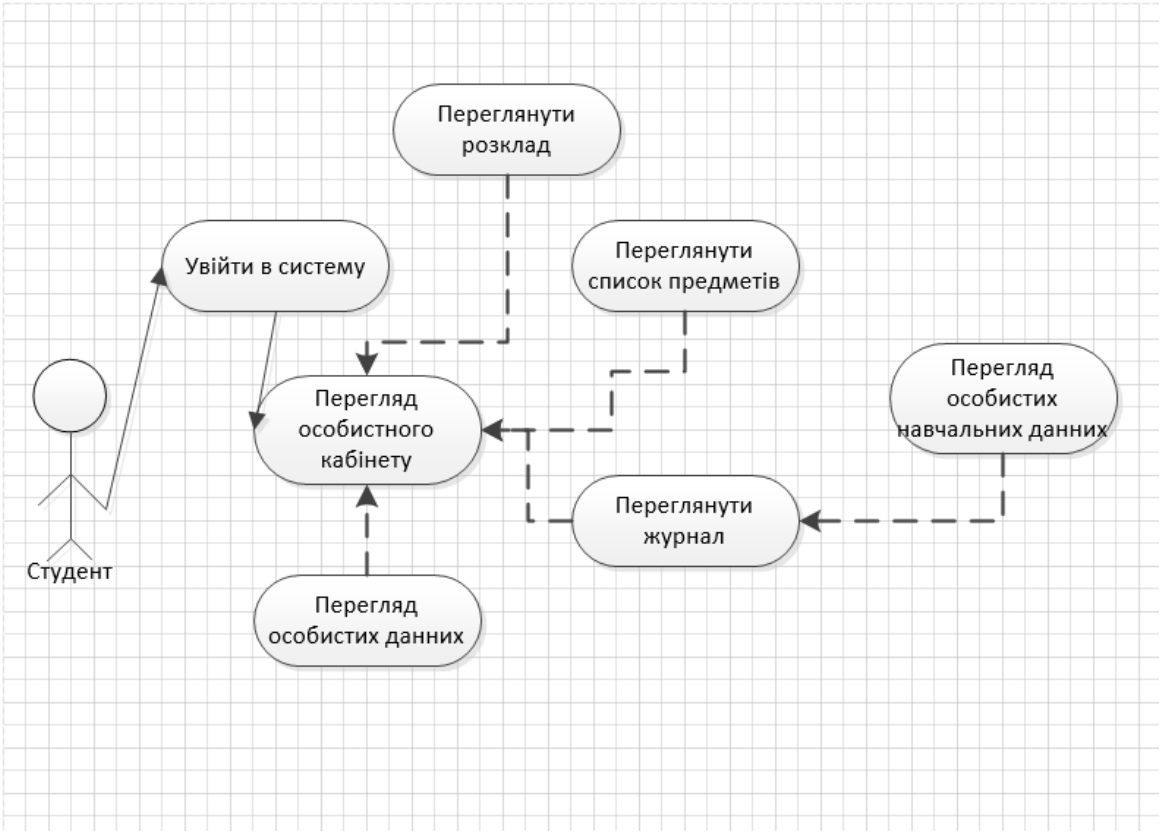


Рисунок 2.2 – Схема ВВ студента

Побудова навігації екранів. Одночасно з виділенням ВВ будується навігація екранів успадкованої системи у вигляді діаграми класів UML. Кожен екран показується в моделі як окремий клас, в якому поля-атрибути відповідають, функціональним кнопкам операції, а кнопкам меню однойменні відносини. Кількість класів повинна відповідати кількості екранів (діалогових вікон).

Рисунок 2.3 – Схема ВВ викладача

Далі проведемо об’єктно-орієнтовний аналіз предметної області та побудуємо діаграми класів проекту.

З проаналізованих даних побудова наступні діаграми: «Діаграма діяльності роботи викладача в журналі», «Діаграма діяльності варіанту використання «Реєстрація», «Діаграма діяльності роботи студента у журналі». Представлено як діаграми діяльності окремих компонентів так і систему в загальному. Дані діаграми представляють послідовність дій користувача в системі, що вони можуть зробити та які наслідки будуть мати їхні дії. Діаграми діяльності є особливою формою діаграм стану, на яких містяться лише діяльності. Діаграми діяльності подібні до процедурних діаграм потоку, але відрізняються від них тим, що діяльності точно прив’язано до об’єктів. Діаграми діяльності завжди пов’язано з класом, операцією або випадком використання.

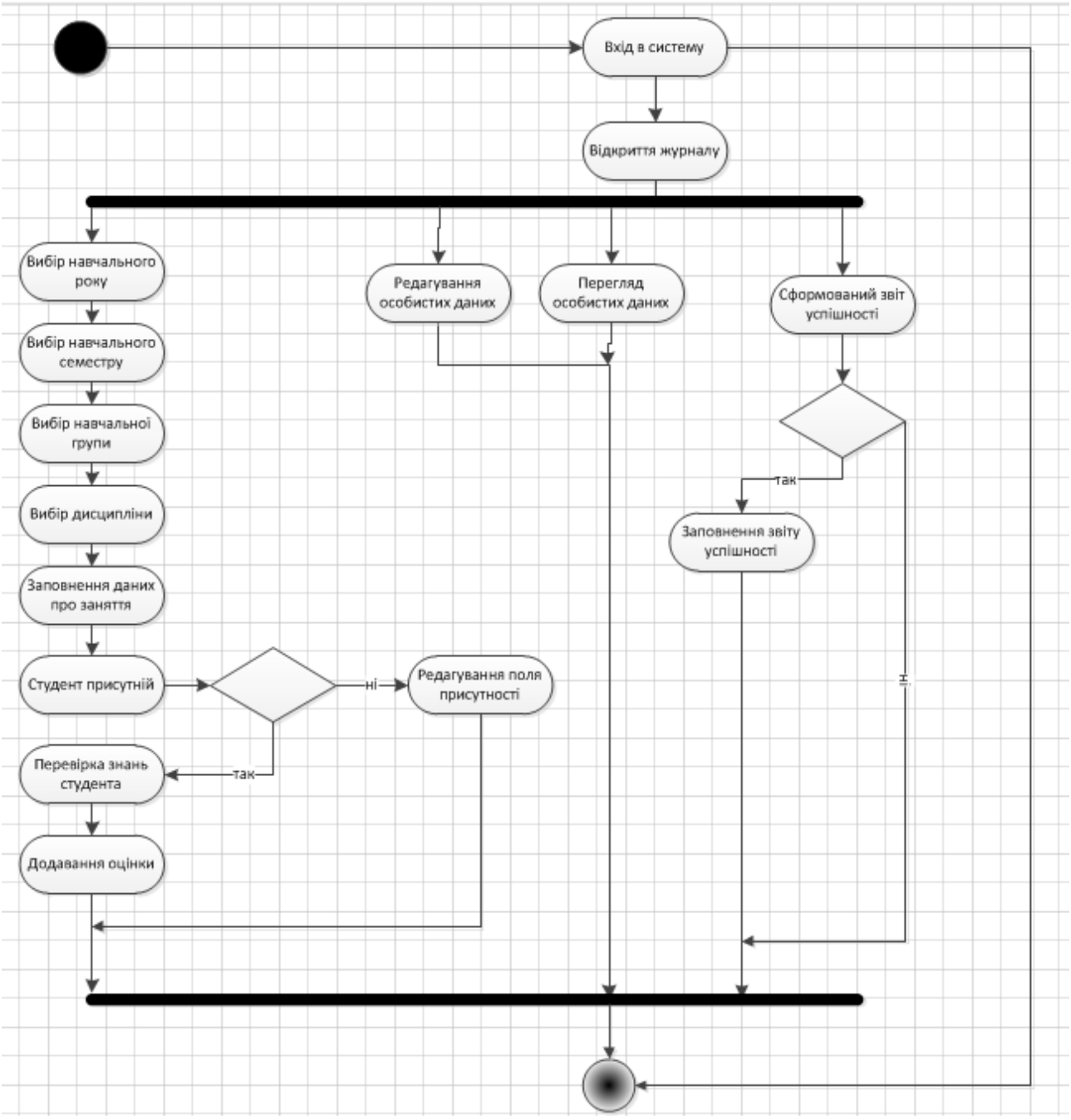


Рисунок 2.4 – Діаграма діяльності роботи викладача в журналі

Також необхідним пунктом є складення варіанту використання системи, при виконанні дії – реєстрації. Які саме дії повинен виконати користувач та яку відповід він може отримати на свої дії представлено на рисунку 2.5.

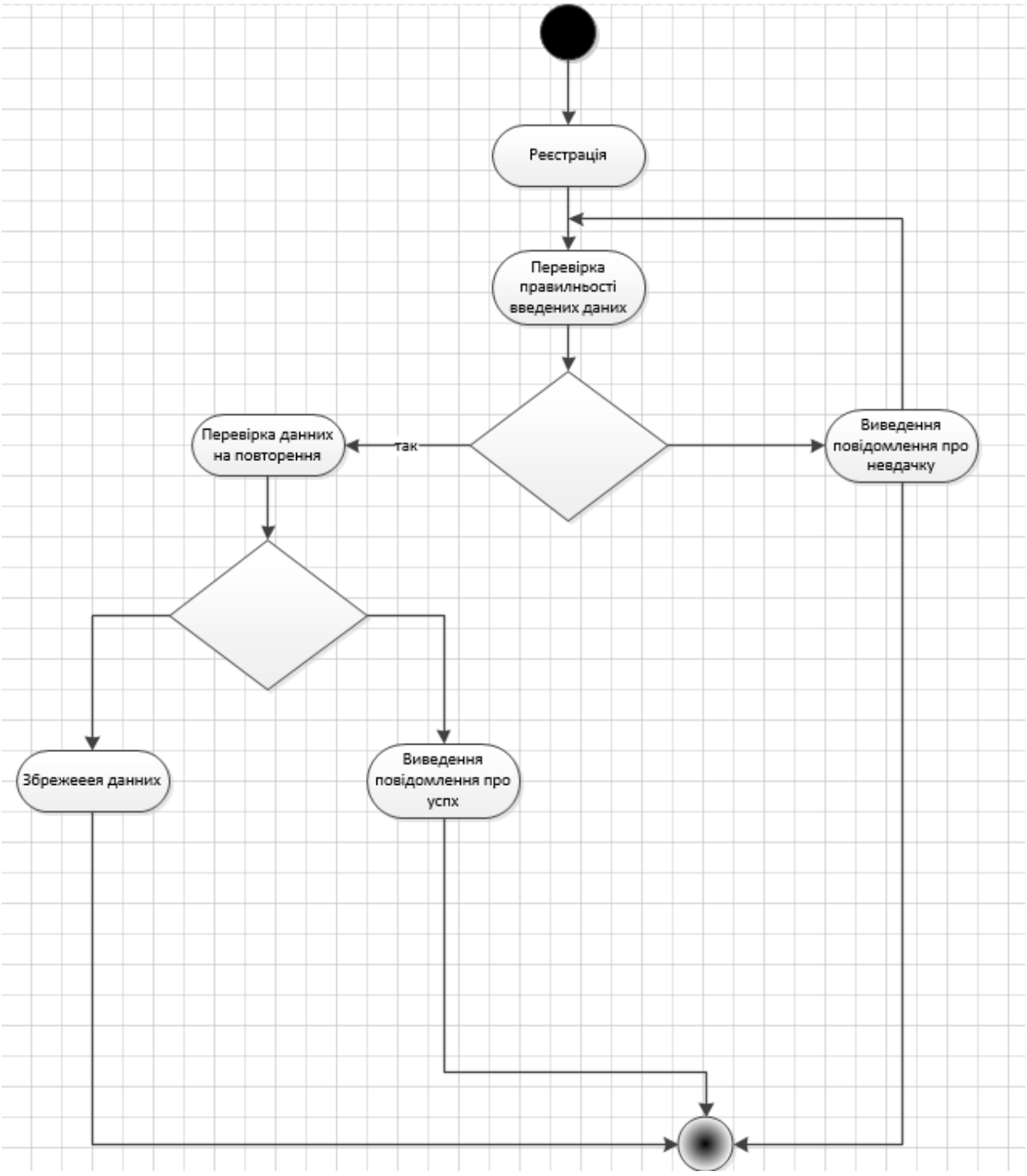


Рисунок 2.5 – Діаграма діяльності варіанту використання «Реєстрація»

Дана діаграма представляє алгоритм дій адміністратора, що буде реєструвати користувача чи викладача. Основні дії це: введення даних користувачем системи, адміністратором у даному випадку, та очікування перевірки правильності їх введення і якщо якісь дані були наверно увведені

системи повідомить про це. Далі неведено діаграму діяльності студента(рисунок 2.6).

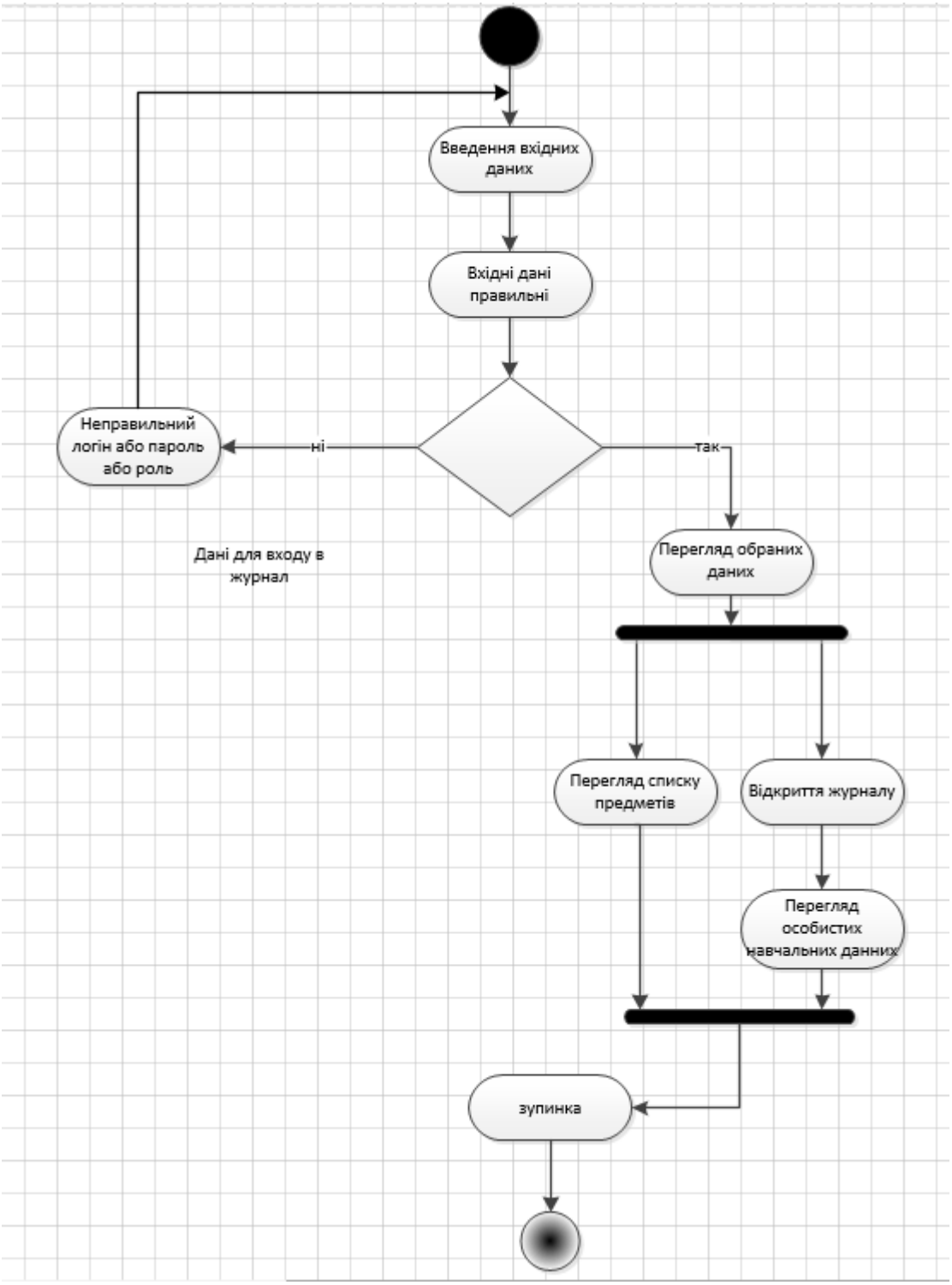


Рисунок 2.6 – Діаграма діяльності роботи студента у журналі

Дана діаграма представляє загальну схему роботи студента в журналі, його можливості та можливу послідовність.

Загальний варіант станів журналу представлено на рисунку 2.7.

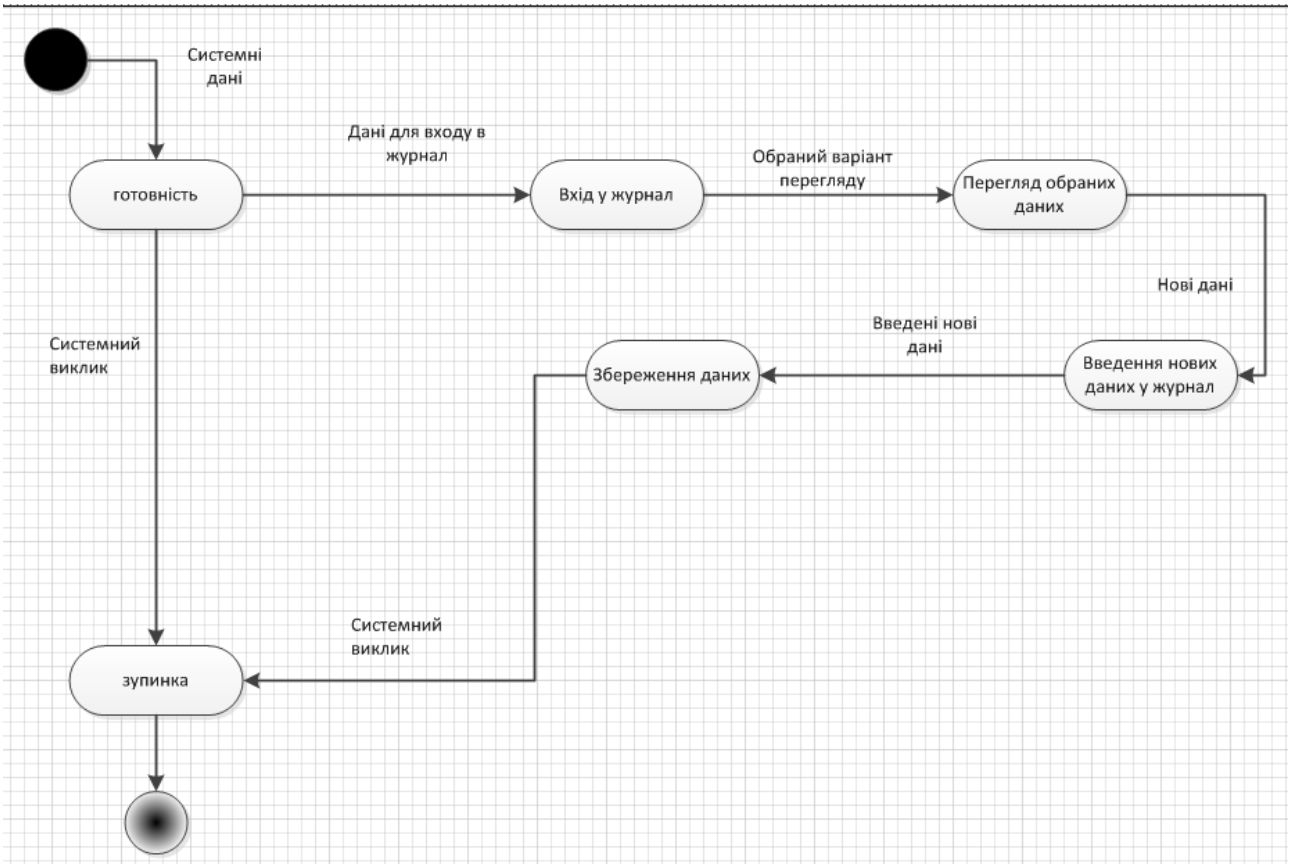


Рисунок 2.7 – Діаграма станів журналу

Виходячи з наведених можна зрозуміти, які самt функції потрібно реалізувати у системі та хто буде користуватися даною системою.

2.2 Розробка алгоритму авторизації та реєстрації

Розроблюваний додаток повинен виконувати такі основні функції: завантаження даних із сервера системи по запиту та відображення отриманих

даних користувачу. Інші функції, викладені у технічному завданні є додатковими і їх виконання вихдить із вищенаведених основних можливостей.

Перша за все для роботи системи її потрібно наповнити данними, а для цього необхідно спочатку зареєструватися відподвіному адміністратору навачльного закладу. Для цього повинен бути продуманий алгоритм реєстрації. Хоч він буде працювати лише на адміністративній частині проте необхідність алгоритму реєстрації присутня оскільки при перегляді списку користувачів при правильній реєстрації можна буде точно ідентифікувати особу університету відповідальну за додавання нових користувачів чи редагуванні даних, що хоть і прошйли перевірку на правильність проте не відповідають дійсності.

Для реєстарції потрібно ввести наступні дані:

- VmRegistrationUserLogin – логін користувача;
- VmRegistrationUserSurname – прізвище користувача;
- VmRegistrationUserFirstname – значення прапорця «запам’ятати мене»;
- VmRegistrationUserMiddlename – ім`я по-батькові користувача, якщо таке наявне;
- VmRegistrationUserBirthdate – дата народження користувача;
- VmRegistrationUserPassword – пароль користувача;
- VmRegistrationUserRole – роль користувача в системі.

Якщо одне з полів незаповнене або невірно заповнене, майбутньому адміністратороів невдасься натиснути кнопку «Підтвердити».

Після заповнення усіх полів у формі реєстрація і натиску на кнопку підтвердити, на сервер відправляються модель яка містить усі ці параметри(рисунок 2.8).

Після цього можна перейти на авторизаційну форму адміністративної частини програми.

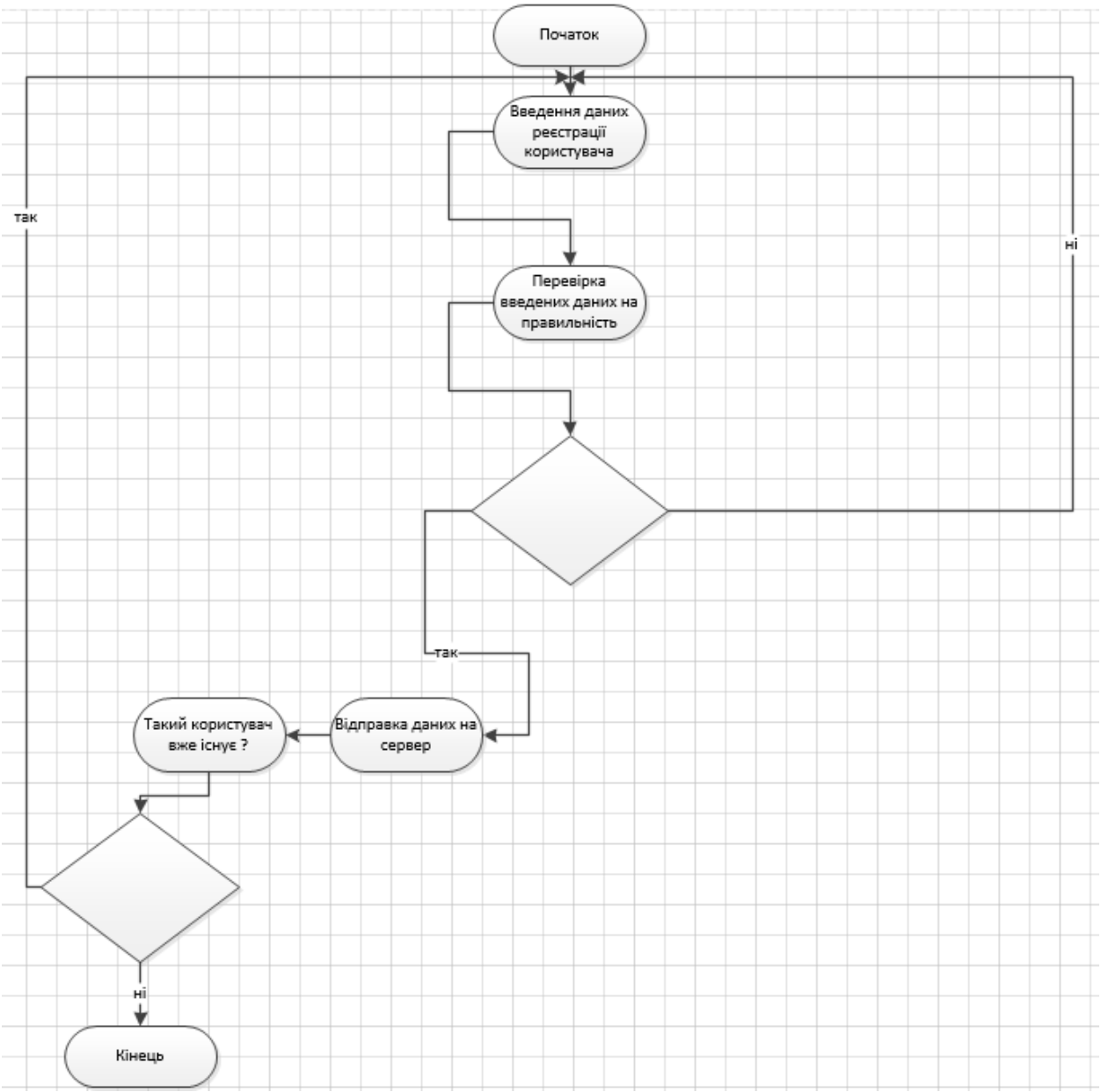


Рисунок 2.8 - Алгоритм реєстрації

Далі пілся успішної реєсстрації адміністратор може авторизуватися. Процес авторизації для адміністратора не вирізняється особливою складністю, тому немає сенсу його представляти.

Проте для авторизації клієнта алгоримт авторизації буде складнішим чим для декстопної адмін частини сервера адміністратора. Як і

адміністратору, користувачу мобільного пристрою потрібно ввести для авторизації наступні поля:

- VmLoginModel[login] – логін користувача;
- VmLoginModel[password] – пароль користувача;
- VmLoginModel [Role] – роль користувача в системі;
- Submit – назва кнопки підтвердження відправки данних ан сервер.

Якщо авторизація пройшла успішно, відбуваються наступні операції: переадресація на сторінку з яка слугує основною сторінкою в системі, з інформацією персональні на навчальні дані студента, якщо це студент, або на сторінку викладача, якшо це викладач, перехід на сторінку списку предметів, перехід на сторінку персонального розкладу та перехід на сторінку журналу. У випадку неуспішної авторизації, переадресації не відбувається, натомість виводиться помилка про те що дані введені користувачем додатку невірні.

Для проходження авторизації через мобільний додаток потрібно пройти ті ж самі кроки, що і при авторизації через адмін частину , але через особливості мобільних додатків, а також заявлених функцій, алгоритм авторизації стає складнішим (рисунок 2.9).

Якщо авторизація пройшла успішно, відбуваються наступні операції: переадресація на сторінку з інформацією про персональні дані та певна частина навчальним данних. У випадку неуспішної авторизації, переадресації не відбувається, натомість виводиться помилка про неправильний логін чи пароль.

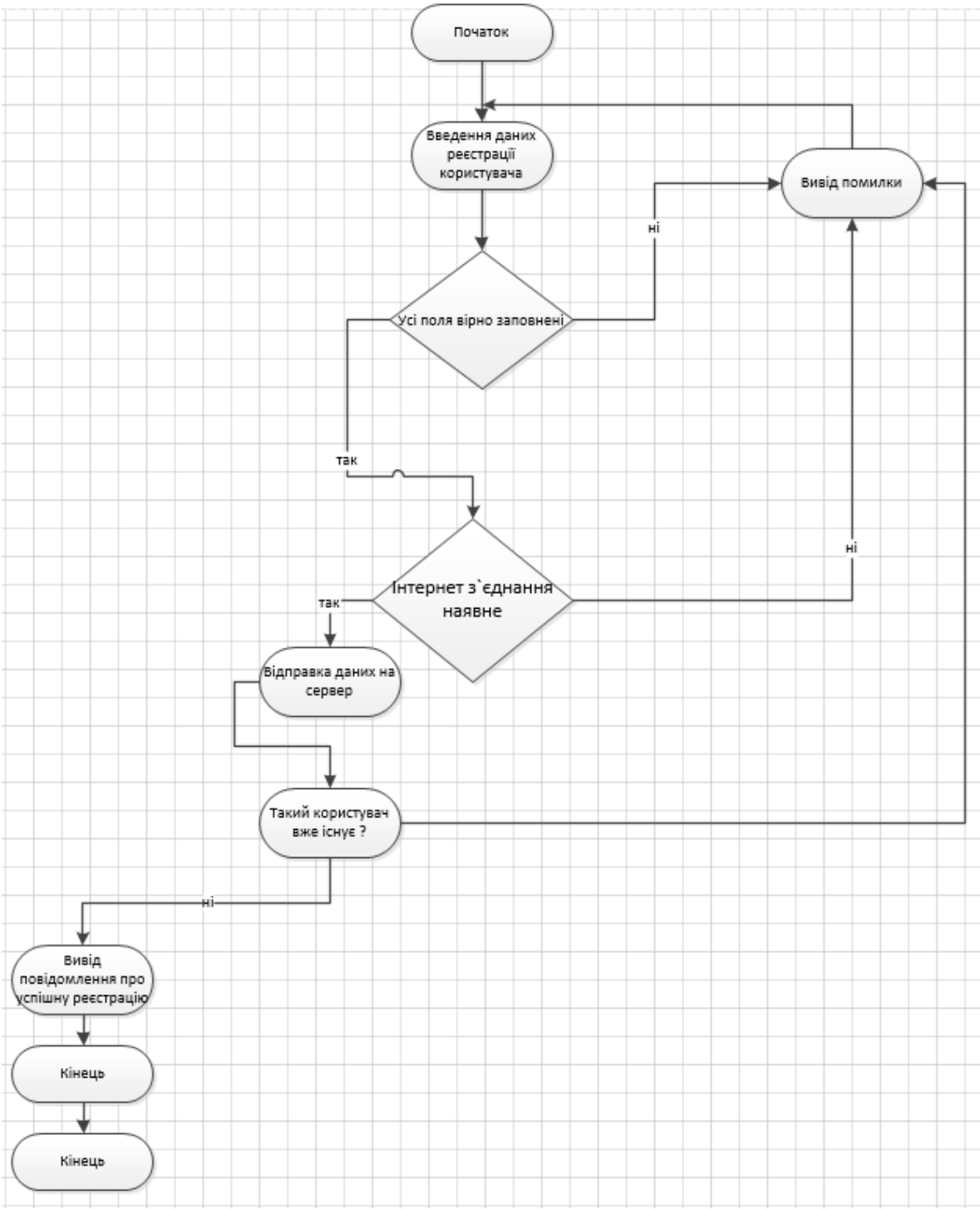


Рисунок 2.9 – Алгоритм авторизації

З даного алгоритму видно, які кроки пройде користувач мобільного додатку у спробі зайти на головне мобільного клієнта журналу.

2.3 Проектування графічного інтерфейсу

Для написання графічної частини, інтерфейсу користувача, як і для самого клієнта використовувся сучасний фреймворк Xamarin Forms. Платформа не має жорстких рекомендацій, як повинні виглядати і працювати додатки. Існує набір рекомендацій, але в основному вони стосуються різного роду елементів типу меню, віджетів та інших речей. У типових мобільних додатках використовується панель дій, поле для навігації, та поле контенту. На даній платформі, можно писати, як і для мобільних пристроїв, що працюють на ОС Android так і для iOS та WindowsPhone. Оскільки Xamarin крос-платформенний фреймворк, пишучи інтерфейс додатку в середовищі розробки Visual Studio, даний інтерфейс буде працювати і на вищезгаданих мобільних операційних системах. Звичайно ж кожно операційна система має свої особливості, тому еелменти написані задопомогою Xamarin будуть відображатися інколи по різному в залежності від системи(рисунок 3.1).

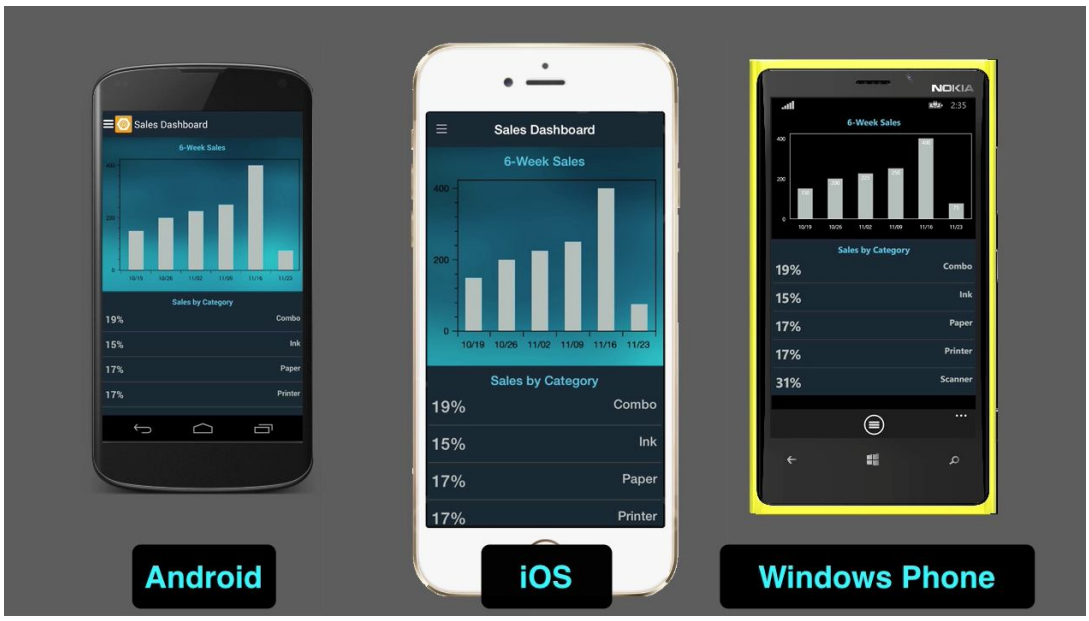


Рисунок 2.10 – Відмінності інтерфейсних елементів побудованих Xamarin для мобільних платформ

Панель управління призначена для розташування на ній іконки додатка, а також основних команд меню, якій найчастіше використовуються. Якщо додаток має декілька вікон, то зазвичай використовується навігаційне поле. Поле навігації викликається з лівого краю екрану, забираючи при цьому деяке місце поля контенту. Поле контенту – це простір, на якому розміщується контент відображений у списку, таблиці, сітці і т.д. Проте неварто хвилюватися, навігаційне поле не закриває контент а лиш займає певне місце зверху і контетн починає відображатися на границі де закінчується навігаційне меню.

Оскільки мобільним додатоком можуть користуватися 2 типи користувачів, варто передбачити різні вигляти вікон для програми. Цього можна досягти двома способами: побудувавши одне вікно і відтворювавши різні елементи на ньому за рахунок можливості приховувати видимість елементів; побудувавши декілька різних вікон для різного роду ситуацій, що можуть виникнути у двох типів користувачів.

Ще однією особливістю є те, що додатком можуть користуватися на різної роздільної здатності дисплеїв. Щоб уникнути проблем з розробкою інтерфейсу для велкої кількості різної родільної здатності дисплеїв, було вирішено більшість вікон зробит лише у портретній орієнтації дисплею. Дане рішення спростить як розробку так і підтримку інтерфейсу. Недоліків небуде оскільки вікна, що вирішено розробити у портретній орієнтації не містять багато інформації і для даних вікон інший тип орієнтації непотрібен.

Для авторизації користувача буде використовуватись один вигляд для всіх типів екрану. Він буде складатись з 3 полів для вводу та кнопки «Підтвердити» (рисунок 2.11).

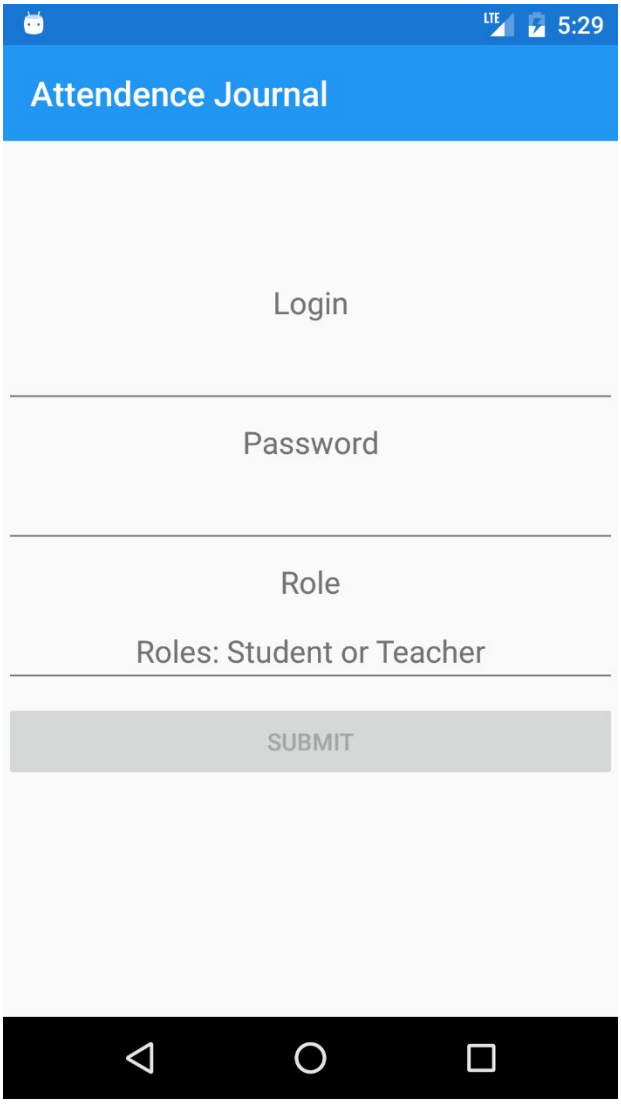


Рисунок 2.11 – Екран авторизації

Наступним представлено головне вікно додатку, в яке буде переходити користувач одразу ж після успішної авторизації або вертаючись з одного із дочірніх вікон системи. Оскільки було вирішено, що в системі будуть 2 типи користувачів і кожен з них буде бачити різну інформацію в основному вікні то необхідно написати для кожного типу користувача своє основне вікно Вікно для типу користувача «Студент» представлено на рисунку 2.12. А вікно для типу користувача «Викладач» на рисунку 2.13.

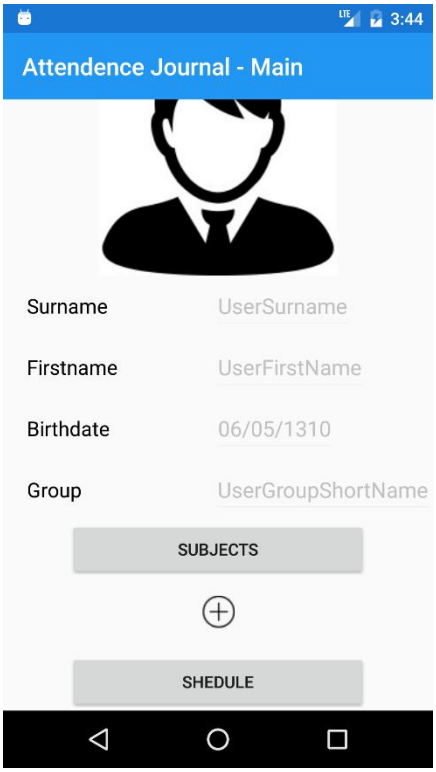


Рисунок 2.12 – Головне Вікно для користувача «Студент»

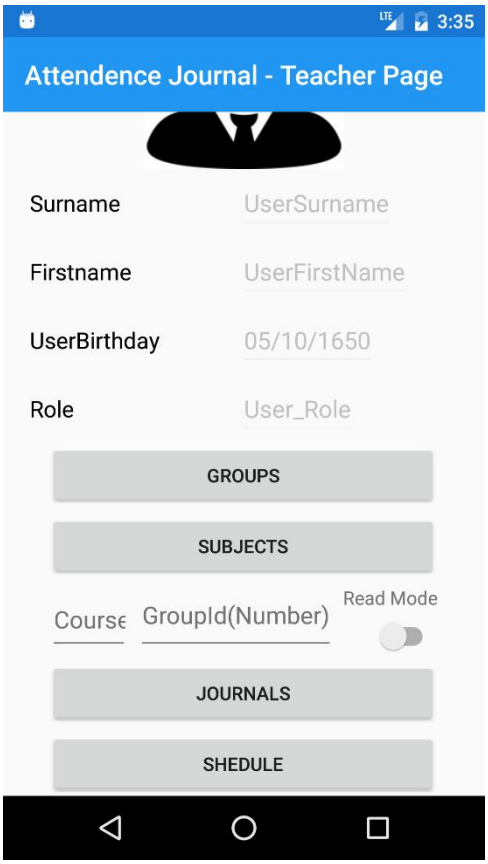


Рисунок 2.13 – Головне Вікно для користувача «Викладач»

Наступним кроком потрібно представити вікно списку предметів до яких залучений користувач, та як, що студент, що викладач бачитимуть однакові поля данних, то потреби в створенні різних вікон для кожного типу користувача відсутня. У даному вікні користувач бачитиме порядковий номер предмета у базі данних, назву предмета та які викладачі закріплені за даним предметом, лектора, практика та опис предмету. Для зручності у списку у полях викладачів будуть відображатися дані викладачі у наступній короткій формі: Прізвище, Імя., наприклад: Бобер Юрій (рисунок 2.14).

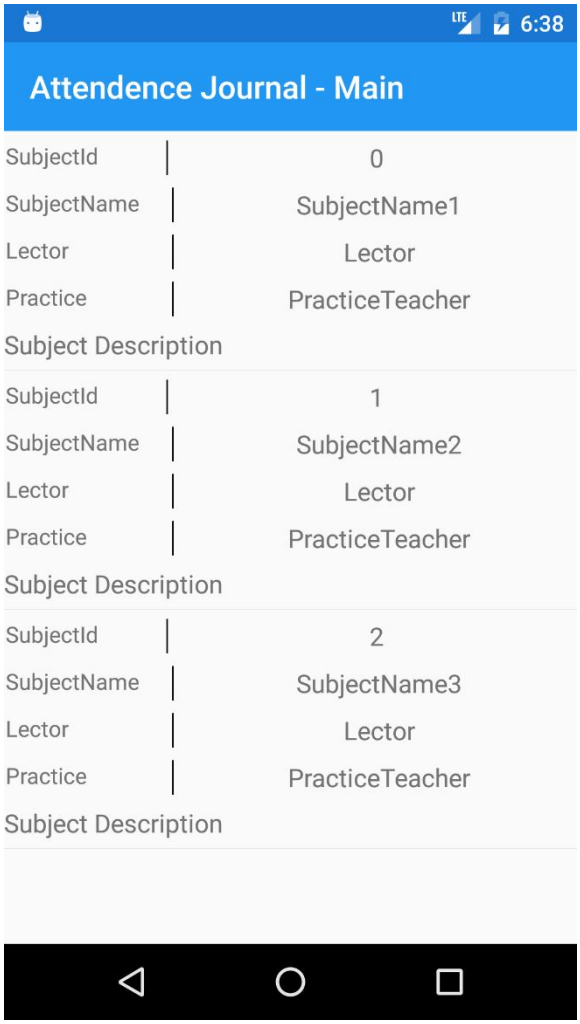


Рисунок 2.14 – Макет списку предметів

Наступним потрібно представити вікно журналу додатку для викладача. Оскільки у викладача більше функцій чим у студента і він може заповнювати журнал, як наприклад: виставляти оцінку, якщо студент був опитаний; ставити відмітку «присутній» або «відсутній», якщо студента небуло на парі. Для встановлення відмітки про пристуність студента на парі був обраний елемент «Xamarin.Forms.Picker»(рисунок).<- PICKER Також варот зазначити, що на сторінці журналу викладача буде показано наступну інформацію: дата проведення заняття, Назва групи в якої проходить заняття та предмет який ведеться на занятті. Варто зазначити, що дана інформація виводиться в шапці списку одразу ж після навігаційного меню(рисунок). Дане вікно краще відкривати в горизонтальній орінтації, проте може бути відкрите і в портретній(рисунок 2.15).

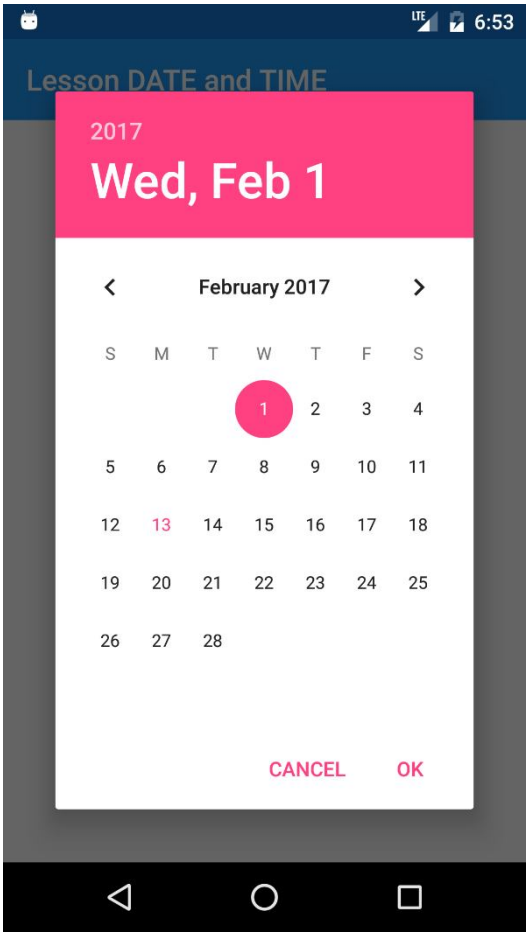


Рисунок 2.15 – Вікно журналу викладача

Наступним представлено вікно додатку журналу користувача, в яке буде переходити користувач одразу ж після введення таких данних, як: ідентифікаційний номер предмету, який можна буде дізнатися у списку предметів, та дати дня за який потрібно переглянути бали. Після цього кнопка «Journals» буде активна та можна буде перейти на відповіднк сторінку. В даному вікні користувач буде бачити список записів журналу за певну дату та відповідно до свого номеру у базі даних. Для вибору дати за якою буде відображатися список записів обрано елемент «Xamarin.Forms.DatePicker»(рисунок 2.16).

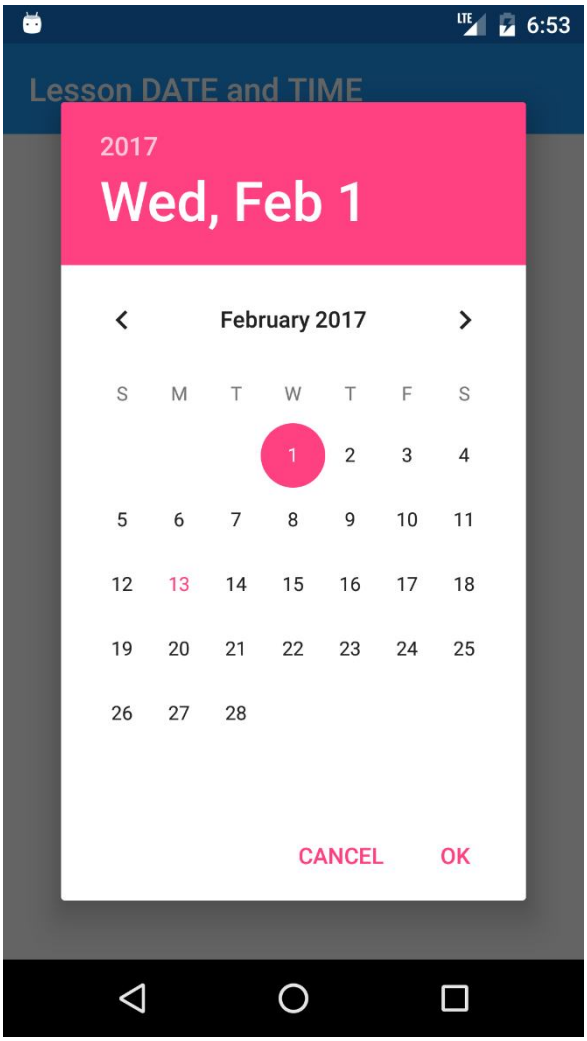


Рисунок 2.16 – Макет вибору дати для додатку

Наступним представлено вікно персонального розкладу користувача. Оскільки у розклад однаковий для будь-яких учасників навчального процесу, тому для всіх типів корисувачів системи буде представлено одне вікно розкладу. Проте варто зазначити, що даний розклад є персональним і зберігаєтсья лише на телефоні користувача. В ньому відображається наступна інморфація: дата коли буде проходити подія, час на яку ця подія призначеня, предмет чи інша подія яка повинна відбутися та лектора предмету чи події. Для встановлення дати використаний елемент «Xamarin.Forms.DatePicker». А для встановлення часу проведення події використано «Xamarin.Forms.TimePicker»(рисунок 2.17). Вікно розкалду представлено на рисунку 2.17.

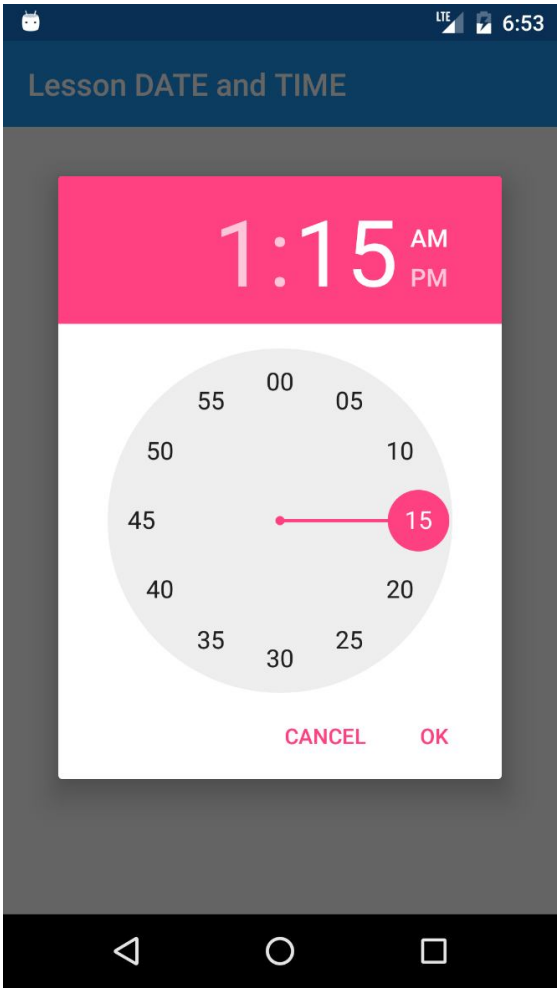


Рисунок 2.17 – Приклад вибору часу у додатку

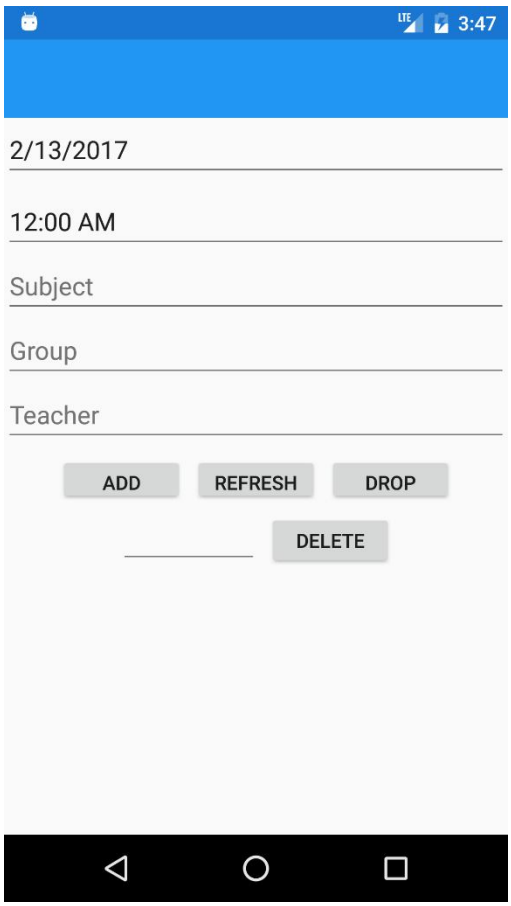


Рисунок 2.18 – Вікно розкладу

Сповіщати користувача про успішність чи не успішність виконання певних дій можна за допомогою спливаючих повідомлень та діалогів. Спливаючі повідомлення у Xamarin, використовуються для показу певної інформації. Діалоги використовуються для інформування користувача про настання певної події, і на відміну від спливаючих повідомлень діалог вимагає певної реакції від користувача. Наприклад, для видалення усіх записів із розкладу використано діалог(рисунок 2.19). Для сповіщення користувача про успішне авторизування чи помилку при вході в систему необхідно використовувати спливаючі повідомлення(рисунок 2.20).

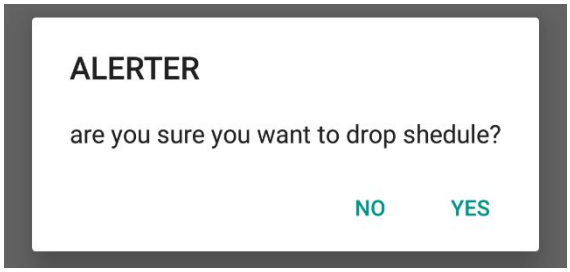


Рисунок 2.19 – Приклад використання діалогу у додатку

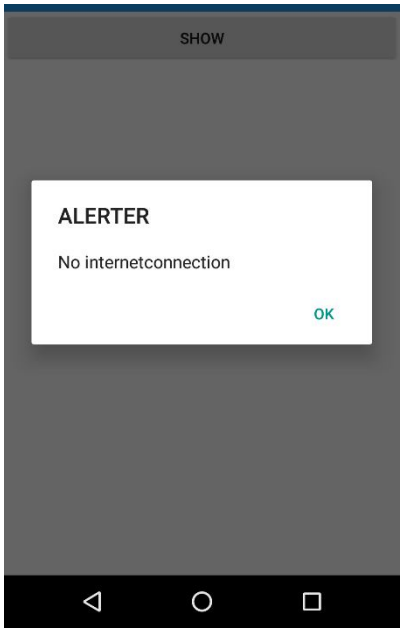


Рисунок 2.20 – Приклад спливаючого повідомлення

З вищенаведених рисунків макетів інтерфейсу користувача мобільного додатку системи, можна точно представити, що саме буде оброблятися в системі та її можливі варіанти використання.

2.4 Проектування сутностей бази данних

Для того, щоб програма прцювала вона повинна кудись зберігати свої дані. Для цього використовують бази данних. База даних (англ. database) –

сукупність даних, організованих відповідно до концепції, яка описує характеристику цих даних і взаємозв'язки між їх елементами; ця сукупність підтримує щонайменше одну з областей застосування. В загальному випадку база даних містить схеми, таблиці, подання, збережені процедури та інші об'єкти. Дані у базі організовують відповідно до моделі організації даних. Таким чином, сучасна база даних, крім саме даних, містить їх опис та може містити засоби для їх обробки.

З визначених сутностей, що будуть взаємодіяти з системою, було визначено що будуть три типи користувачів: студент, викладач та адміністратор. Дані типи користувачів будуть представені в сутності «Користувач». Було вирішено не створювати 3 таблиці для кожної сутності а використату одну універсальну. Дані викладача та студента у даній система майже не відразняються тому це не створить незрозумілості чи хаосу коли буде переглядатися списоку користувачів. Отже, для користувача було обрано наступні поля, що будуть зберігатися у базі(рисунок 2.21):

- ідентифікаційний номер користувача, що буде генеруватися базою данних власноруч та буде збільшуватися по мірі запису данних;
- прізвище користувач;
- Ім`я користувач;
- Ім`я по-батькові, якщо таке присутнє;
- Пароль користувача;
- Дата народження користувач;
- Роль користувача в системі;
- Група до якої належить користувач.

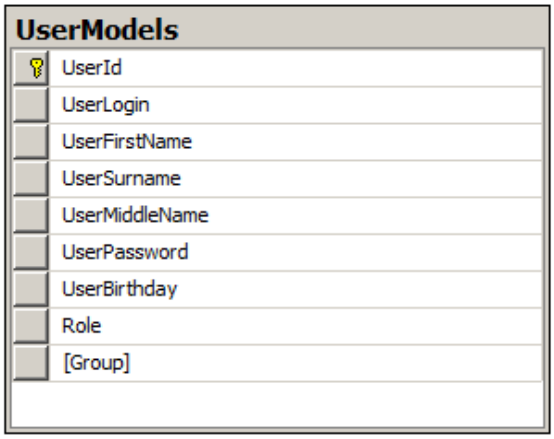


Рисунок 2.21– Таблиця «Користувач»

Група в даній таблиці є посилання на запис у іншій таблиці «Група», що означає для бази даних, що це є зовнішній ключ. Також оскільки авторизація відбувається по логіну, то дане поле в базі даних помічене, як унікальне. Це значить наступне: у кожного користувача повинен бути унівальний від інших користувачів логін.

Наступною буде сутність «Група. В даній таблиці будуть зберігатися дані, що дадуть змогу ідентифікувати групу. В ній будуть зберігатися наступні дані(рисунок 2.22):

- Ідентифікатор групи; назва групи;
- Номер курсу до якого належить група;
- Номер групи якшо таких груп є більше чим одна;
- Назва групи.

В даній таблиці немає посилань на інші таблиці.

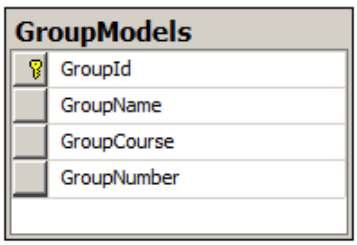


Рисунок 2.22 – таблиця «Група»

Наступна сутність «Предмет». Дана таблиця використовується для збереження предметів, що викладаються у вищому навчальному закладі. Так, як у кожного предмета є певні характеристики, як назва, лектор і тому подібне то вони і будуть записані і таблицю, а саме(рисунок 2.23):

- Ідентифікатор предмету;
- Назва предмету;
- Лектор, що закріплений за предметом;
- Викладач практичних занять закріплений за предметом;
- Опис предмету.

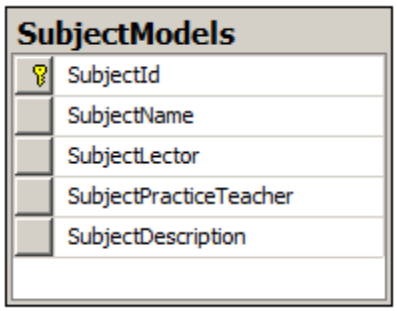


Рисунок 2.23 – Таблиця «Предмет»

Останньою сутністю, буде таблиця «Журнал». Дана таблиця і буде основним з місць робити викладача чи студента. Тому в дану сутність було виділено наступні поля(рисунок 2.24):

- Ідентифікатор запису журналу;
- Група, яка в якої проводиться заняття(посилання на запис в таблиці «Група»;
- Користувач, в якого проводиться заняття(посилання на запис в табилці «Користувач»;
- Дата проведення заняття;
- Відмітка за заняття студенту, оцінка або відмітка пропуску чи присутності;

- Предмет, що ведеться в групи і студента(посилання на запис в групі «Предмети».

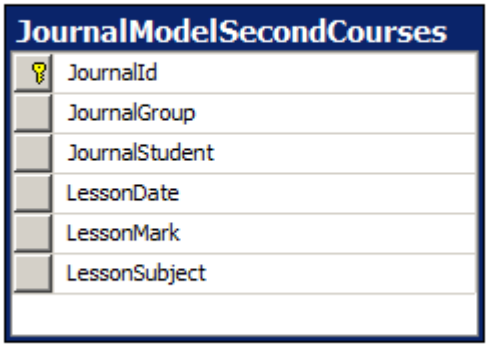


Рисунок 2.24 – Таблиця «Журнал»

З даних сутностей можна чітко зрозуміти, що буде зберігатися і базі данних та для чого дане сховище даних може бути використане. Також потрібно вказати зв`язки між даними таблицями.

Оскільки користувач може бути у майже у всіх таблицях і може бути у різних записа виещзгаданих таблиць то зв'язок таблиці «Користувач» з іншими таблицями крім «Група» встановлено «один-до-багатьох».

В таблиці «Предмет» є посилання на користувачів проте для кожного запису таблиці «Група» ідентифікуєтсья один користувач, хоча в 1 рядку групи є поля які посилаються на користувача. Тому зв`язок «один-до-одного».

В таблиці «Журнал» є посилання на користувача, на групу та на предмет, і можна сказати, що одинму запису з групи завжди буде відповідати один користувач, одна група та один предмет. З стороини таблиці «журнал» зв`язок до таблиці на які посилаються буде «один-до-одного», проте з іншої сторони зв`язок буде "один-до-багатьох» так, як один користувач може бути у багатьох записах таблиці «журнал». Загальну структуру бази даних представлено у додатку В.

Варто також зазначити, що в цілях ефективності роботи сервера та економії пам`яті, таблиця «журнал» поділена га п`ять таблиць. Кожна таблиця відповідає за свій курс у вищому навчальному закладі.

2.5 Обґрунтування вибору інструментарію розробки

Для розробки мобільної системи системи управління навчальним процесом було використані наступні інструменти:

- Об`єктно-орієнтовану мову c#;
- Vusial Studio 2015 Community Edition;
- Entity Framework;
- Xamarin Forms framework;
- Windows Presentation Foundation (WPF) - система для побудови клієнтських додатків Windows з візуально привабливими можливостями взаємодії з користувачем, графічна підсистема у складі .NET Framework, яка використовує мову XAML.

C# (вимовляється як Сі-шарп) – одна із об'єктно-орієнтованих мов програмування, що містить надійну та безпечну систему типізації для платформи .NET. Синтаксис C# близький до C++ і Java. Перша за все C# розроблялась як мова програмування прикладного рівня для CLR. Тому вона в своїй основі залежить, від можливостей самої CLR. CLR в даному випадку це, якщо перекладати безпосередньо, «Загальне середовище виконання мов» (Common Language Runtime). Дане середовище дозволяє програмістам не відволікатися на деякі особливості деталів про конкретний процесор, на якому буде використовуватися програма.

Переваги мови C#:

- Середній поріг навичок для практичного входження людей, що знайомі з якою-небудь мовою, схожою на C і відповідно не є дуже легким для тих, хто з такими мовами взагалі незнайомий.
- Все є під ООП, навіть елементарні типи даних.
- JIT-компіляція виконується відразу в команди необхідної для роботи програми архітектури.
- Величезна кількість вже готових класів на всі випадки життя, тільки й чекають, щоб ними скористалися.
- Для мови є власна середовище розробки.
- Присутня можливість працювати з пам'яттю безпосередньо.
- Є служба «Microsoft .NET Framework NGEN», яка відразу компілює в натівний бінар і кешує цей бінарник для його наступних запусків.
- Інтеграція з некерованими мовами однією командою (зокрема з C і C ++). Потрібні функції Кернела? Легко!.

Xamarin - це фреймворк для кроссплатформенной розробки мобільних додатків (iOS, Android, Windows Phone) з використанням мови C #. Ідея дуже проста. Ви пишете код на своєму улюбленому мовою, із застосуванням всіх звичних для вас мовних особливостей нібито LINQ, лямбда-виразів, Generic`ов і async`ов. При цьому ви маєте повний доступ до всіх можливостей SDK платформи і рідного механізму створення UI, отримуючи на виході додаток, яке, строго кажучи, нічим не відрізняється від рідних і не поступається їм у продуктивності.

Фреймворк складається з декількох основних частин:

- Xamarin.IOS - бібліотека класів для C #, що надає розробнику доступ до iOS SDK;
- Xamarin.Android - бібліотека класів для C #, що надає розробнику доступ до Android SDK;
- Компілятори для iOS і Android;

- Плагін для Visual Studio.

Переваги Xamarin.Forms:

- По-перше, це всім нам добре знайомий C # і .NET. Якщо ви давно вже пишіть на Шарп, то вам не треба витрачати багато часу на вивчення кількох нових фреймворків, а то і мов. Ну або, принаймні, на початку не треба, і ви можете досить швидко стартанути, використовуючи свої поточні знання.

- По-друге, підхід до створення і роботи з призначеним для користувача інтерфейсом близький до того, до чого ми всі звикли в Windows. Особливо раді будуть розробники WPF, так як Xamarin Forms підтримує роботу з XAML, Біндінг, темплейти, стилі і інші радощі життя. Думаю, зрозуміло, що вони кілька урізані і не варто очікувати всієї потужності WPF, але все-таки зручності це додає.

- Так як це C #, то наступний плюс в тому, що можна повторно використовувати вже написаний код. У більшій частині він буде працювати коректно. Є у платформ обмеження, але вони не настільки великі. У нас вийшло завести досить великий шматок з XtraGridControl-a, і це нам дуже допомогло.

- З того, що Xamarin.Forms схожий з WPF, випливає наступний плюс цієї платформи: MVVM. Дійсно, Xamarin.Forms має XAML, візуальні елементи мають BindingContext (аналог DataContext в WPF), є BindableProperty (аналог DependencyProperty). Таким чином, можна пов'язувати View з ViewModel аналогічно тому, як в WPF.

- Ще одна перевага даної платформи в тому, що так як UI описується тільки в одному місці, то додатки під різними системами будуть виглядати дуже схоже. Що може бути важливо, наприклад, в корпоративних розробках.Деякий час назад досить широку популярність здобули ряд фреймворків (наприклад PhoneGap), які пропонують розробку кроссплатформенних мобільних додатків на HTML5 з використанням

JavaScript. Ідея полягає в тому, що програма розробляється як звичайний сайт для мобільних пристроїв з використанням відповідних js-бібліотек, наприклад, Jquery Mobile. Потім все це пакується в якийсь контейнер, який для користувача виглядає як нативное додаток. Мінуси цих фреймворків очевидні:

- по-перше, ви не маєте доступу до нативним елементам UI. Тобто навіть якщо ви хочете використовувати стандартну кнопку «Назад» для iPhone, ви повинні її намалювати і зверстати.
- По-друге, ви отримуєте урізаний і узагальнений API для роботи з платформою. Таким чином, ті чи інші фічі, властиві якоїсь окремої платформі будуть вам недоступні. Ну і третє і найважливіше - такий додаток фізично запускається всередині браузера телефону (точніше всередині контрола WebView). Не потрібно розписувати довго, що це означає: низька продуктивність (особливо «хороший» WebView на старих версіях Android) і величезні проблеми з відображенням (ну, панове, це ж - браузер). Хоча, звичайно, в певних випадках ці фреймворки можуть виявитися дуже доречні.

Microsoft Visual Studio - засіб для розробників ПЗ, яке дозволяє вирішувати основні завдання розробки: система спрощує створення, налагодження та розгортання додатків на різних платформах, включаючи SharePoint і хмарну середу.

Основними перевагами Visual Studio є:

- Використання обчислювальних потужностей локального комп'ютера і хмари;
- Проста реалізація спільних завдань та індивідуальний підхід;
- Функція підтримки декількох моніторів.

ВИСНОВКИ ДО ІІ РОЗДІЛУ

У другому розділі проведено проектування основних елементів системи «мобільна система управління навчальним процесом у ВНЗ», а саме:

- проаналізувавши основні взяємодіючи елементи, такі як: актори, можливості системи було спроектовано загальну структуру системи. Виявлено, що користуватися системою будуть студенти та викладачі вищого навчального закладу. Представлено можливі варіанти використання системи виявленим акторам. Визначено основні функції доступні як системі в цілому так і користувачам;
- завдяки аналізу мобільних платформ та аналогів розроблено алгоритм реєстрації, що використовується в адміністративній частині сервера та при додаванні нового корристувача до системи. Також розроблено алгоримт авторизації для мобільного клієнта, що ускладнюється особливостями мобільної платформи, а саме: крім перевірки введених даних потрітбно також перевіряти наявність інтернет з`єднання, після чого очікується відповідь від сервера чи такий користувач існує;
- Спроектовано авторизаційне вікно; кореневе вікно де представлено персональна та деяка навчальна інформація; вікно, що передставляє список предметів до яких залучений користувач; вікно журналу, де буде працювати викладач чи переглядати студент(по 1 вікна для користувача «студент» та «викладач» , оскільки мають різні права на роботу в журналі; вікно персонального розкладу.
- згідно акторів та пункту призначення системи спроектовано сутності бази даних та надано зв`зяки між ними. В результаті спроектовано 4 сутності: «Користувач», спільна таблиця для всіх типів користувачів, «Група», «Предмет» та «Журнал».

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОБІЛЬНОЇ СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМ ПРОЦЕСОМ У ВНЗ

3.1 Розробка архітетури системи

Діаграмою класів в сфері UML називається діаграма, на якій зображено набір класів, а також зв'язків між цими класами. Крім того, діаграма класів може містити коментарі і обмеження. Обмеження можуть неформально задаватися на звичайній мові або ж можуть формулюватись на мові об'єктних обмежень OCL. На діаграмах класів відображаються також властивості класів та методи.

Розроблюваної системи для управління навчальним процесом містить 41 клас, кожен з яких має своє призначення, а саме:

- ****Model – призначений для опису моделей системи, таких як: корисутвач, група, предмет, журнал, логін;
- IRepository – інтерфейс, що описує основні методи якими повинен повинен володіти клас для взаємодії з базою даних;
- *****Repository – класи, що реалізовують інтерфейс IRepository та виконують вибірку даних з таблиць, в якому реалізовано основні команди мови маніпуляції данним(вибірка, видалення, запис, обновлення);
- EncryptDecryptPassword – клас, що використовується для шифрування та дешифрування паролю. Даний клас реалізовує шифрування паролю алгоритмами MD5 та TripleDES;
- DataContext – клас призначений для ініціалізації база даних та різного її роду налаштувань;
- *****Controller – клас, що описує вхідн точки сервера, керує запитами користувача (одержувані у вигляді запитів HTTP GET або POST, коли користувач натискає на елементи інтерфейсу для виконання різних дій). Його

основна функція - викликати і координувати дію необхідних ресурсів і об'єктів, потрібних для виконання дій, що задаються користувачем. Зазвичай контролер викликає відповідну модель для задачі;

- UntiOfWork – клас, призначений для управління контроллерами, щоб всі вони них мав єдиний контекст роботи з базою даних

- MapConfigfile – клас, призачений для створення можливих варіантів перетворення одних об`єктів в інші, в даному випадук перетворює сутності бази даних в іншу модель в якої є хоча б одне поле з сутності-джерела;

- CheckInputObject – клас призначений для перевірки валідності введених користувачем даних згідно заданої бізнес-логіки;

- Commands – клас, призначений для реалізації інтерфейсу ICommand, який потрібен команда-методів згідно концепції архітетури MVVM;

- BoolToVisibleConverter – клас, призначений для перетворення вхідного булевого значення в опцію видимості UI-елемента(User Interface element);

- ViewModelManager – основний клас адміністративної частини сервера, призачнений для опрацювання подій, що відбуваються на UI-частині та відповідно на них реагувати виконуючи команди до моделі чи надсилаючи інформацію на UI-частину;

- MainViewModelManager – основний клас мобільної частини сервера, призачнений для опрацювання подій, що відбуваються на UI-частині та відповідно на них реагувати виконуючи команди до моделі чи надсилаючи інформацію на UI-частину;

- App – клас, призначений для запуску стартової сторінки мобільного додатку та обробки подій програм мобільних пристроїв таких, як: onresume, onsleep та on OnStart;

- ShedulePage – клас призначнений для генерації SQlite бази даних додатку, що працює в місці збереження програми. Для кожної мобільної операційної системи місце збергіння інше і визначається особливостями системи;

- SheduleModel – клас призначений для опису моделі журналу для мобілного додатку;
- ISQLite – інтерфейс призначений для реалізації в класах мобільних платформ отримання шляху до збереженого файлу бази даних SQLite;
- StartViewPage – клас, що містить ініціалізатор стартової сторінки мобільного додатку;
- AJRootPage – клас, що містить ініціалізатор основної сторінки сторінки мобільного додатку;
- JournalPage – клас, що містить ініціалізатор сторінки журналу мобільного додатку для типу користувача «викладач»;
- StudentJournalPage – клас, що містить ініціалізатор сторінки журналу мобільного додатку для типу користувача «студент»;
- SubjectPage – клас, що містить ініціалізатор сторінки мобільного додатку призаченї для відображення списку предметів користувача;
- StartView – клас, що містить ініціалізатор стартового вікна адміністративної частини сервера;
- AttendenceJournalMain – клас, що містить ініціалізатор основного вікна адміністративної частини сервера;
- AttendenceJournal-LogIn – клас, що містить ініціалізатор вікна авторизації адміністративної частини сервера.

3.2 Реалізація ядра сервера

Для реалізації ядра сервера використовувався відомий фрейморк для роботи з базою даних на мові C# Entity Framework. Entity Framework являє спеціальну об'єктно-орієнтовану технологію на базі фреймворка .NET для

роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то Entity Framework являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, але на концептуальному рівні, який нам пропонує Entity Framework, ми вже працює з об'єктами.

Перша за все потрібно створити моделі об`єктів, що будуть зберігатися в зберігатися в базі даних. Для цього даний фреймворк пропонує 3 шляхи якими можна досягти створення БД:

- Database first: Entity Framework створює набір класів, які відображають модель конкретної бази даних

- Model first: спочатку розробник створює модель бази даних, по якій потім Entity Framework створює реальну базу даних на сервері.

- Code first: розробник створює клас моделі даних, які будуть зберігатися в бд, а потім Entity Framework за цією моделлю генерує базу даних і її таблиці

Для даного проекту було обрано Code First підхід, в якому як вже зазначалося будуються класи моделі даних з яких буде створена сама база даних.

Створюємо звичайний public клас в якому описуємо відповідні поля які повинні бути в даній моделі даних. Після цього потрібно вказати фреймворку, яке поле буде яким. Поля можуть бути просто звичайними полями таблиці, можуть бути посилання на запис в іншій таблиці, тобто зовнішніми ключами, можуть бути первинного ключами, що використовуються базод даних для ідентифікації кортежу відношень, тобто ідентифікує певний рядок в таблиці з значенням в полі первинного ключа. Даний ключ є автоікрементним тому його непотрібно вводити в модельі при записі даних. Також .NET фреймворк пропонує такий простір імен, як

«DataAnotations», який містить в собі клас для налаштування полів класу, наприклад для надання налаштувань полям бази даних. До цих налаштувань відноситься: «Required» атрибут, що означає що дане поле є обов`язковим і неможе бути незаповненим. При записі в БД моделі в якої єє незаповнене поле «Required» виникне помилку запису, осільки поле неможу бути 0; атрибут «StringLengh()t», що означає що дане поле буде містити ту кількість символів яка задана цифрою в дужках даного атрибути; атрибут «Key», що дає зрозуміти Enityty Framework, що дане поле буде первинним ключем в даній таблиці; атрибут «Foreign Key», повідомляє фреймворк, що дане поле буде посилання на інший запис в іншій таблиці; атрибут «Index()», всередині якого є властивість isUnique, що дозволяє зробити дані в полі визначеним цим атрибутом як унікальні, тобто при записі в поле з даним атрибутом такожо ж запису призведе до помилки. Для полів моделі «Користувач» було визначено використано наступні атрибути до полів:

- Первинний ключ(Primary Key) – UserId;
- Index(isUnque) – Login;
- ForeignKey(зовнішній ключ) - для поля Group, що дає зрозуміти, що в цьому полі посилання на запис в таблиці «Група».

Також оскільки в даній системі більше ніж одна модель і як вищезгадувалося існують посилання на записи в інших таблиця. Для цього потрібно знати, який зв`язок між таблицями буде існувати, якщо «один-до-одного» то достатньо лиш вказати зовнішній ключ для поля яке буде містити посилання. Для зв`язку «один-до-багатьох» клас, який може міститися в багатьох записах іншої таблиці повинен визначити в своєму класі властивість типу «ICollection<ваш_тип> ім`я змінної {get; set;} та проініціалізувати її в базовому конструкторі класу. Якщо потрібно реалізувати зв`язок «багато-до-багатьох» то потрібно в класах які підтримують цей зв`язок визначити по властивості типу «ICollection<ваш_тип> ім`я змінної {get; set;} і як було

вищезгадано порініціалізувати її в конструкторі. В результаті клас «користувач» на діаграма буде мати наступний вигляд(рисунок 3.1).

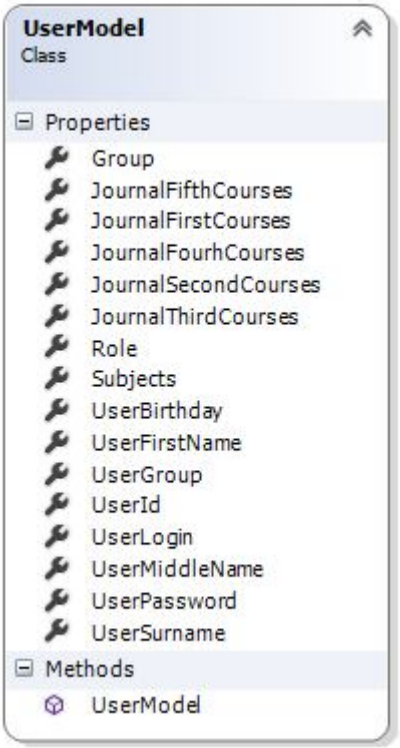


Рисунок 3.1 – Клас «користувач» створений Entity Framework

Аналогічним чином будуємо інші моделі та класи для розроблюваної системи управління навчальним процесом у ВНЗ.

Проте це не все. Entity Fremework незможе з цього побудувати базу даних з таблицями. Для побудови таблиць БД потрібно реалізувати ще один найважливіший клас, який буде наслідуватися від класу Entity Framework DbContext. Розробники Entity Framework вирішили зробити більш простий інтерфейс для роботи з даними, уклавши в нього всі можливості ObjectContext, але при цьому полегшивши рішення різних завдань для роботи з даними, для яких тепер ви можете використовувати стандартні шаблони. Новий набір класів був запроваджений у версії Entity Framework 4.1.

Головними класами нової спрощеної платформи є DbContext, DbSet і DbQuery, а інтерфейс для роботи з цими новими класами називають DbContext API. Даний клас є базовим в Entity Framework і надає широкі можливості по роботі з базою даних: створення запитів, відстеження змін і збереження даних в базі. Для цього в класі що буде наслідуватися від нього в стандтартному конструкторі потрібно передати в базовий конструктор ім`я змінно, яка містить стрічку з інформацією про параметри які необхідні, щоб отримати з`єднання з БД. Також в конструкторі класу можна добавати таку команду як: Database.SetInitializer<назва _класу>(), та викликати всередині нього команду «CreateDatabaseIfNotExists<назва _класу>. Це дозволить створити вашу БД при ініціалізації даного класу, якщо вона до цього небула створена.

Наступним кроком є власне створення сутностей БД з ващественних класів. Це досягається за допомогою DbSet<>. DbSet описує набір сутнісних класів, який потім можна використовувати в коді для створення запитів CRUD (створити, прочитати, обновити, видалити) до даних. За допомогою реалізації відповідного класу описуються різні об'єкти бази даних (таблиці, представлення, збережені процедури і т.д.). Дане встановлення виглядє, як звичайна властивість класу.

Також при наслідування від DbContext отримуємо ще один сильний інструмент для роботи з базою даних – ModelBuilder. Цей клас є зв`язувачем моделі - він створює зв'язок між класами моделі і схеми бази даних. DbContext дозволяє взаємодіяти з цим класом завдяки методу OnModelCreating (), в якому ви можете внести налаштування прив'язки моделі, перед її побудовою. Такий підхід до налаштування моделі називають Fluent API. Встановлюємо DbSet для усіх моделей і база даних може бути створена.

Після створення БД потрібно реалізувати управління організацією запитів до БД. Для цього можна використати популярний для цих цілий паттер «Репозиторій». Він дозволяє абстрагуватися від конкретних підключень до джерел даних, з якими працює програма, і є проміжною ланкою між класами, безпосередньо взаємодіють з даними, і решті програмою. Наприклад, якщо БД створена в MS SQL Server або задопомогою MongoDB, прийшлося б писати багато коду для адаптування, а даний паттерн дозволяє з мінімальними зусиллями уникнути цього. Данний паттерн представляє собою інтерфейс, в якому описуються методи, які система повинна буде виконувати з БД. Для БД розроблюваної системи було визначено наступні операції:

- GetItemsList – отримати список записів таблиці;
- GetItembyId – отримати запис вказавши в полі його ідентифікатор;
- AddItem – додати запис в таблицю;
- UpdateItem – обновити запис в таблиці;
- RemoveItem – видалити запис з таблиці.

Реалізовується він доволі просто, дані функції пишуться задопмогою звичайного C# коду накшталт: _(змінна_типу_Datacontext)._назва_таблиці._ функція_C#(умова вибірки).

Проте тепер виникає проблема при якій, оскільки даний сервер реалізовує MVC модель, яка говорить що повинен бути контроллер, що буде контролювати роботу з таблицею БД, потрібно створити багато класів репозиторіїв оскільки багато сутностей і це призводить до ускладнення роботи і можливості того, що різні контроллер будуть використовувати різні контексти данних. Патерн Unit of Work дозволяє спростити роботу з різними репозиторіями і дає впевненість, що все репозиторії використовуватимуть один і той же контекст даних.

Оскільки сервер буде приймати веб-запити потрібно створити точки входу запитів на наш сервер. Тепер згідно ASP.NET MVC необхідно створити контроллер для кожного репозиторія, які будуть приймати веб-запити від клієнтів даного сервера. ASP.NET MVC являє собою платформу для створення сайтів і веб-додатків з використанням патерну MVC.

Контролер (controller) представляє клас, з якого власне і починається робота програми. Цей клас забезпечує зв'язок між моделлю і представленням. Отримуючи вводяться користувачем дані, контролер виходячи з внутрішньої логіки при необхідності звертається до моделі і генерує відповідне подання.

Розроблюваний сервер немає представлення, оскільки він виступає, як веб-сервіс. Також варто зазначити, що рівень доступу до бази даних(репозиторії та unitofwork) як і бізнес-рівень(контроллери) не проводить ніяких обчислень лише CRUD операції тому він підпадає під концепцію REST веб-сервіса.

REST - це архітектурний стиль, побудований на існуючих, добре відомих і контрольованих консорціумом W3C стандартах, таких, як HTTP, URI, XML. У REST-сервісах акцент зроблений на доступ до ресурсів, а не на виконання віддалених сервісів. Приклад REST запиту представлено на рисунку 3.2.

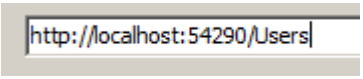


Рисунок 3.2 – REST запит на отримання списку користувачів

Реалізація контроллера відповідає методам закладеним в інтерфейсі IRepository. В ньому створюється екзепмляр класу unitofwork який виступає як точка через яку будуть отримані запити будуть виконувати відповідні репозиторій-функції.

Для того, щоб контроллер зміг приймати HTTP-запити потрібно використати HTTP-атрибути, що визначені в просторі імен System.Web.http.

Дані атрибути дозволяють описати тип HTTP-запиту, щоб буде оброблятися даним методом та також при створенні ASP.NET MVC контроллера обов`язковим є встановлення типу, що повертатиме метод як «IHttpActionResult», що автоматично створить HttpResponseMessage відповідь та запакує відповідь користувачу у HTTP-запит. Програмісту непотрібно робити це самому, за нього це робить asp.net mvc фреймворк. Відповідь даного методу може будь-яка згідно визначених методом відповідей. Зазвичай це наступні відповіді: OK(код статус 200) – запит користувача оброблено і все завершилося успішно; BadRequest(код статус 400) – невірний синтаксис в запиті. Також оскільки в нас не один а декілька контроллерів кожен з яких відповідає за роботу з конкретними даними потрібно організувати маршрутисзацію запитів до севрвера. Це досягається задопомогою Route-атрибутів. Вони дозволяюьб ідентифікувати контроллер чи функції в контроллері з певним ім`ям(рисунок 3.3).

```
[HttpGet]
[Route]
public IHttpActionResult GetUsersList()
{
    var listusers = _unitofwork.User.GetItemsList();
    var allusers = Mapper.Map<IEnumerable<UserModel>, ICollection<BllUserModel>>(listusers);
    return Ok(allusers);
}
```

Рисунок 3.3 – Вигляд налаштування метода в контроллері

Як видно з рисунка даний метод не містить ніяких параметрів в Route-атрибуті, це означає, що якщо ввести пов`язане з контроллером ім`я і більше нічого то запит користувача буде направлений в маршрут по-замовчуванню. Що в даному випадку виступає метод GetUsersList. Клас-контроллер для робити з запитами направленими на роботу з моделлю користувача представлено на рисунку 3.4.

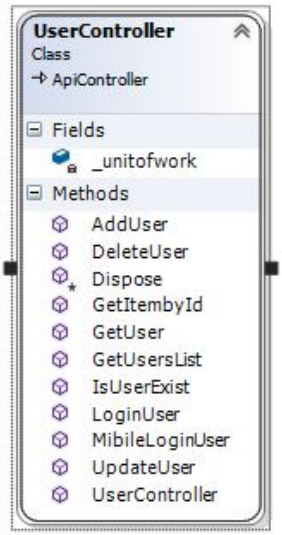


Рисунок 3.4 – методи класу usercontroller

Основна аспекти реалізації сервера для розроблюваної системи полягають в повторенні вище написаного для кожної моделі в системі. Це дозволяє створити REST веб-сервіс, що непотребує інтерфейсу і може видавати дані, як і будь-який інший сервіс.

3.3 Реалізація адміністративної частини сервера

Для реалізації адмінстративної частини сервера використовувався .NET Framework використовуючи одну із його техноголій, а саме WPF. Технологія WPF (Windows Presentation Foundation) є частина екосистеми платформи .NET і являє собою підсистему для побудови графічних інтерфейсів.

Якщо при створенні традиційних додатків на основі WinForms за отрисовку елементів управління і графіки відповідали такі частини ОС Windows, як User32 і GDI +, то додатки WPF засновані на DirectX. У цьому полягає ключова особливість рендеринга графіки в WPF: використовуючи WPF,

значна частина роботи по відображенні графіки, як найпростіших кнопочок, так і складних 3D-моделей, лягати на графічний процесор на відеокарті, що також дозволяє скористатися апаратним прискоренням графіки.

Однією з важливих особливостей є використання мови декларативною розмітки інтерфейсу XAML, заснованого на XML: ви можете створювати насичений графічний інтерфейс, використовуючи або декларативне оголошення інтерфейсу, або код на керованих мовах C # і VB.NET, або поєднувати і те, і інше.

Потрібно зауважити, що задопомогою XAML пишеться сам інтерфейса частина адмінки сервера.

Розробка адміністративна частина розпочнеться з вікна авторизації. Оскільки для авторизації потрібно ввести лиш логі, пароль та вибрати роль то для побудови інтерфейсу в xaml використаємо наступні елементи(рисунок 3.5):

- Label – які будуть використовуватися для того, щоб користувач міг ідентифікувати поле вводу з місце для введення данних;

- Textbox – використовується для введення користувачем даних;

- Combobox – вкористовується для вибору ролі з випадаючого списку;

- Button – використовується для підтвердження відправки даних на рівень доступу до база даних.

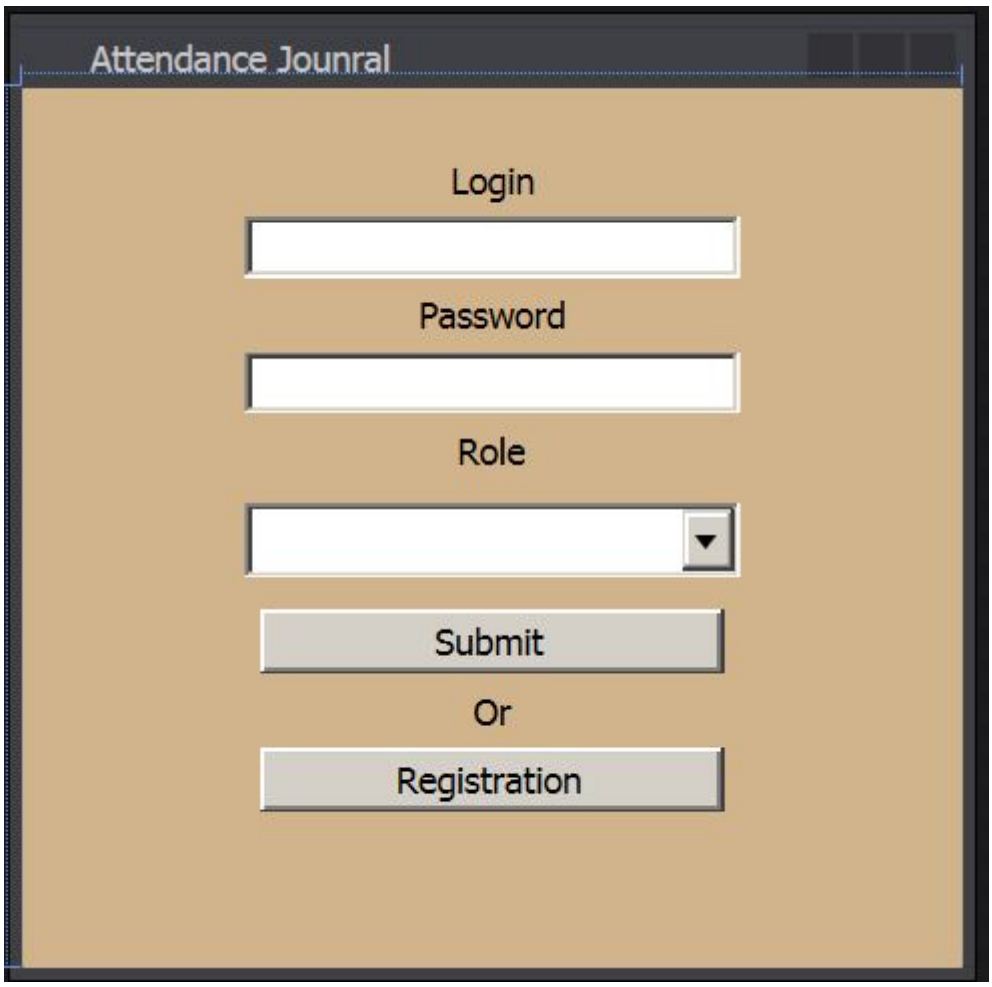


Рисунок 3.5 – Стартове вікно адмін частини

Варто зазначити, що кнопка «Submi» є недоступною для натискання поки усі поля небудуть заповнені і заповнені дані не будуть відповідати бізнес-правилам закладеним в програму. При натисканні на кнопку «Registration» відкриється вікно реєстрації користувача у систему. Попри це вікно авторизації залишиться відкритим, щоб після вдалої реєстрації, можна було закравши реєстраційне вікно розпочати вхід в систему.

Наступним реалізуємо реєстраційне вікно системи. Для реєстрації потрібне ввести дані визначені у моделі користувача. Для створення реєстраційного вікна використано ті ж елементи WPF-технології, що і для авторизації. Добавлено лиш зображення поряд із блоком реєстрації, для надання вікну кращого вигляду. Варто зауважити, що якщо задопомогою

даної програми буде здійснюватися реєстрація викладача, то поле «Group», яке призначене більше для користувача буде заблоковано і небуде змоги його ввести(рисунок 3.6).

Рисунок 3.6 – Вікно реєстрації користувача в систему

Далі реалізуємо основне вікно адміністратора в якому буде здійснюватися його робота. Для цього потрібно буде вже використати більше елементів. Для побудови головного вікна адміністратора використано наступні елементів(рисунок 3.7):

- Listview – в якому виводяться дані з таблиць;
- Menu – для відображення стандатрних можливостей будь-якого вікна а саме: підменю «Завершити програму» та «підменю» довідка.
- TabControl – який дозволяє переключатися між відображенням різни сутностей БД не переходячи на нове вікно. В кожний табконтрол вкладено listview для відображення списків данних

- Також добавлено бокову панель у вигляді StackPanel в середині якої, знаходяться елементи взаємодії з системою. Для вкладки «Користувачі» бокова панель буде відображати поля данних моделі «Користувач», також наявні три кнопки, що дозволяють додати користувача, видалити чи обновити його дані.

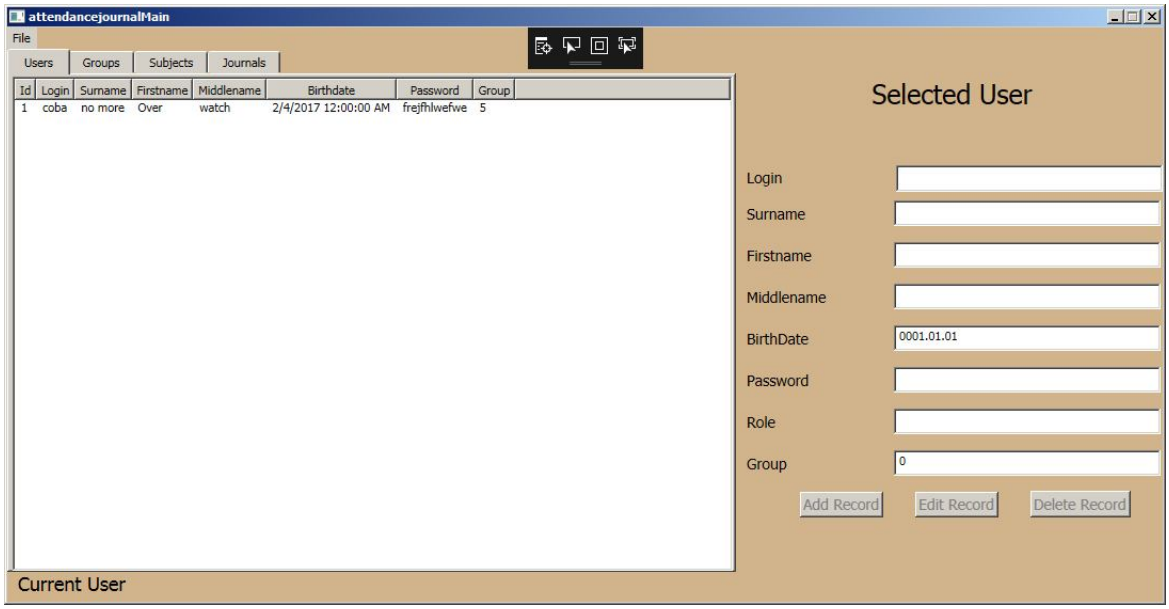


Рисунок 3.7 – Головне вікно адміністратора

Інтерфейс адмін панелі розроблено. Потрібно розробити серцевину. Для розробки функціональної частини даної системи було обрано паттерн MVVM, який дозволяє відділити UI-частину від бізнес-логіки).

Модель описує використовувані в додатку дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління.

Перегляд або подання визначає візуальний інтерфейс, через який користувач взаємодіє з додатком. Стосовно до WPF уявлення - це код в

XAML, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) в WPF може містити як інтерфейс в XAML, так і прив'язаний до нього код C #, проте в ідеалі код C # не повинен містити якийсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна , Вся ж основна логіка додатки виноситься в компонент ViewModel.

ViewModel або модель представлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна даних, що відображаються в поданні, хоча безпосередньо модель і уявлення не пов'язані.

ViewModel також містить логіку по отриманню даних з моделі, які потім передаються в уявлення. І також VewModel визначає логіку по оновленню даних в моделі.

Отже перш за все створюємо клас, який буде містити нашу модель-представлення. Імплементовуємо INotifyPropertyChanged інтерфейс, що дозволить представленню мати інформацію коли потрібно себе обновити.

Далі створюємо приватні змінні тих же типів що і моделі так проініціалізовуємо їх в конструкторі. Тепер для написаної реалізації інтерфейсу потрібно публічні властивості для доступу представлення до даних приватних змінних. У полі «set» властивостей викликаємо імплементоваий метод інтерфейсу. Взагалі як тільки ми щось змінюємо де б це небуло потрібно викликати метод onproppertychanged() оскільки це повідомить представлення, що потрібно обновити даний елемент чи змінну(рисунок 3.8).

```
public string VmLogin
{
    get { return _loginvm.Login; }
    set
    {
        if (_loginvm.Login != value)
        {
            _loginvm.Login = value;
            OnPropertyChanged(nameof(VmLogin));
        }
    }
}
```

Рисунок 3.8– Використання INotifyPropertyChanged згідно MVVM

Так потрібно зробити для кожної властивості, яка так чи інакше буде відображатися на представленні та яку потрібно буде обновляти.

Іншою важливою частиною MVVM є використання команд. Це не означає, що зовсім не можемо використовувати події і подієву модель, проте всюди, де можливо, замість подій слід використовувати команди. Оскільки це залишає чистим code behind сторінки і дозволяє програмі бути легкопідтримуваною. Для використання команд потрібно реалізувати інтерфейс ICommand.

Загальна сигнатура команди наступна: Команда(Метод що виконається, метод який повертає булеву змінну на основі певної умови).

Залежно від того що поверне canexecute метод(той що повертає булеву змінну) і залежить чи зможемо ми виконати команду чи ні. Повертаючись до попереднього прикладу реалізаіції представлення головного вікна адміністратора. Кнока «додати користувача» як до цього кнопки «підтвердити» на вікнах реєстрації та логіну, не буде активна до тих пір поки необхідні дані не будуть введені. Аналогічно кнопки «редагувати» та «видалити» не будуть активні поки небуде здійснен вибір користувача(рисунок). Це якраз і досягається використанням команд. Можна

сказати, що модель-представлення це набір властивостей та команд, які виконують певні дії.

Варто згадати і третій потужний елемент WPF а саме механізм прив`вязки данних Binding. У WPF прив'язка (binding) є потужним інструментом програмування, без якого не обходиться жодне серйозне додаток.

Прив'язка на увазі взаємодію двох об'єктів: джерела і приймача. Об'єкт-приймач створює прив'язку до певного властивості об'єкта-джерела. У разі модифікації об'єкта-джерела, об'єкт-приймач також буде модифікований
Для визначення прив'язки використовується вираз типу: {Binding ElementName = імя_об'екта-джерела, Path = Свойство_об'екта-джерела}. Ключовим об'єктом при створенні прив'язки є об'єкт System.Windows.Data.Binding. Якщо в подальшому нам стане не потрібна прив'язка, то ми можемо скористатися класом BindingOperations і його методами ClearBinding () (видаляє одну прив'язку) і ClearAllBindings () (видаляє всі прив'язки для даного елемента). Варто зазначити, що прив`язка може здійснюватися не лише до властивостей моделі-представлення, а також і до інших елементів представлення, це досягається задопомогою властивості класу Binding: ElementName. Наступною важливою властивістю є Mode. Він встановлює режим зв`язування між джерелом на приймачем. Даний режим може мати наступні значення: один раз(здійснюєтсья при ініціалізації), в однму напрямку(від джерела до приймача), в обида шляхи(зміна буде відбуватися як зі сторони прийамача так і з сторони джерела) та останній режим в одному напрямку до джерела(обновлятися буде тільки змінна джерела). Ще однією цікавою властивістю є UpdateSourceTrigger. Одностороння прив'язка від джерела до приймача практично миттєво змінює властивість приймача. Але якщо ми використовуємо двосторонню прив'язку у випадку з текстовими полями (як в прикладі вище), то при зміні приймача

властивість джерела не змінюється миттєво. Так, в прикладі вище, щоб текстове поле-джерело змінилося, нам треба перевести фокус з текстового поля-приймача. І в даному випадку в справу вступає властивість UpdateSourceTrigger класу Binding, яке задає, як буде присходят оновлення. Це властивість як приймає одне із значень перерахування UpdateSourceTrigger: PropertyChanged: джерело прив'язки оновлюється відразу після поновлення властивості в приймальнику. Приклад використання механізму прив`язок можна було споглянути на попердніх рисунках інтерфейсу адміністратора і також на рисунку.

Важливо також знати, шо сам механізм незнає де знаходяться властивості моделі-представлення, що прив`язати до них певні UI-елементи. Де буде знаходитися змінна прив`язки можна вказати задопомогою властивості DataContext, RelativeSource чи Soure. Як вищезгадувалося прив`язувати UI-елементи можна не тільки змінних чи інший інтферйсних елементів, а також до команд, що власно і дозволяє командам бути написаними в моделі-представленні(рисунок 3.9).

```
<Menu x:Name="ManagerMenu" HorizontalAlignment="Left" Height="22" VerticalAlignment="Top" Width="Auto">
    <MenuItem Header="File" Height="Auto">
        <MenuItem Header="Close" Command="{Binding CloseApplictionCommand}"/>
        <MenuItem Header="Help" Height="Auto" Command="{Binding HelpCommand}"/>
    </MenuItem>
```

Рисунок 3.9 – Використання механізму прив`язок на команду

Ще одним з основних аспектів даної частини сервера є те, що розроблена адміністративний додаток працює поряд з ядром сервера, тобто вона не надсилає запити на контроллери і чекає відповіді, тобто працює безпосередньо з проміжним рівнем бази данних, а саме з репозиторіями через паттерн UnitOfWork.

3.4 Реалізація мобільного додатку системи

Для реалізації мобільного клієнта системи використовуєтсья сучасний і перспективний фреймворк Xamarin Forms. Також як і для адміністративної частини використовувався паттерн MVVM.

При створенні проекту потрібно вибрати між двома типами проектів в Xamarin: Portable і Shared. Blank App (Xamarin.Forms Portable):

- даний шаблон ствоює декілька проектів, один з яких пзиначений для створення загальної dll бібліотеки зі всією логікою. А інші проекти для різни мобільних платформ викиорстовують дану dll;
- Blank App (Xamarin.Forms Shared): даний шаблон также створює декілька проектів по 1 для кожної операційної системи ОС. Але тепер всі вони використовують набір загальних файлів.

Обираємо Xamarin Portable. Створивши у 2 розділі інтерфейси тепер потрібно написати їх взаємодію та принципи роботи. Стартове вікно містить 3 поля введення даних та 3 звиачних тектсових поля для відображення призначення даних які потрібно відрпавити в поле під текстом. Аналогічно WPF додатку, потрібно прив`язати дані елементи ло властивостей, що знаходиться в моделі-представленні.

Також передбачно перевірку полів на наявність в них введених даних, для того, щоб можна було відправити дані тільки після їх введення. Проте навідміну MVVM, команди тут дещо інакші і доступність клавіші потрібне прив`язувати на іншу функцію, що так аналогічно командів повертає true чи flalse. Після чого можна натиснути кнопку відправки, якщо усі поля вірно заповнені. Настинувши кнопку шар що містить поля для вводу даних блокується і запускається ActivityInidcator, що призеачений в даному випадку

демонстрації користувачеві, що його заапит відправлено на сервер необхідно очікувати відповіді(рисунок 3.10).

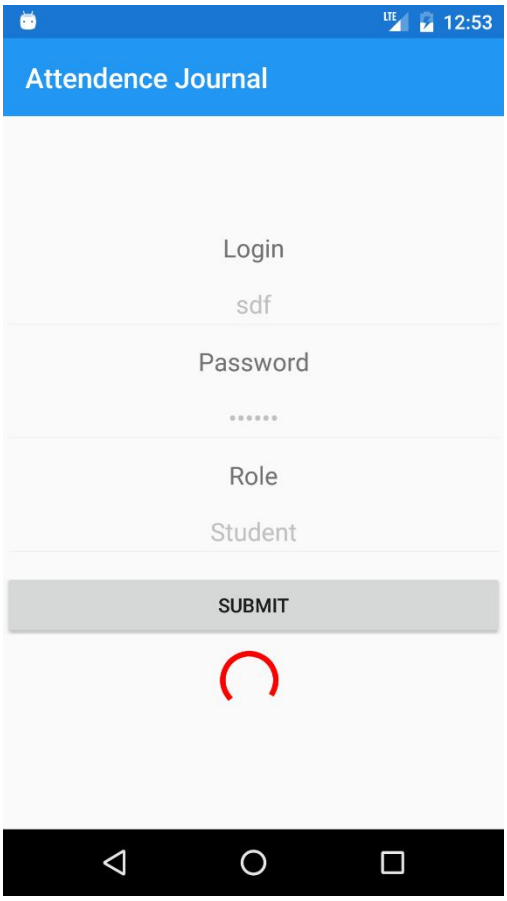


Рисунок 3.10 – Процес авторизації

Для того, щоб додаток міг відправляти http запити на сервер необхідно реалізувати створення httpclient, здійснити серіалізацію в json формат, оскільки сервер відправляє дані саме в такому форматі та здійснити надсилання. Для цього було встановлено плагін Rest.Client, який вже має це все реалізовано в собі потрібно лиш скористатися(рисунок 3.11).

```
public async Task<List<T>> GetAsync()
{
    var httpClient = new HttpClient();

    var json = await httpClient.GetStringAsync(WebServiceUrl);

    var taskModels = JsonConvert.DeserializeObject<List<T>>(json);

    return taskModels;
}
```

Рисунок 3.11 – Приклад форми запиту до веб-сервісу

Пілся отримання відповіді від сервіса виникне спливаюче вікно, що повідомить користувача чи була успішна реєстрація. Якщо успішна, то користувача буде перенаправлено на головну сторінку журналу. Якщо ні буде можливість відправити запит знову і також в повідомленні буде вказано причину невдачі: невірний пароль, невірний логін, невірна роль , тощо.

Для збереження персонального розкладу розкладу користувача використано SQLite, полегшену базу даних для зберігання даних. Дана БД являється встроєною в додаток, що означає що взаємодія додатку з SQLite не схоже нв взаємодію клієнта з сервером. Вона стаю частиною програми. Дані занесені у розклад зберігаються у файлі впросторі програми або відповідно до операційних систем у відведених для цього місцях.

ВИСНОВКИ ДО ІІІ РОЗДІЛУ

У третьому розділі здійснено реалізацію основних частин системи «мобільна система управління навчальним процесом у ВНЗ», а саме:

- проаналізувавши основні взяємодіючи елементи, побудован систему класів, які наявні в системі, описано їх функціонал та вміст. Представлено, який клас за що відповідає та з ким взаємодіє. На основі цього можна чітко зрозуміти,

який клас до якої частини програми належить, коли буде виконуватися та де, яку відповідальність на нього покладнено.;

- на основі розробленої архітектури, аналізу моделей, що будуть в системі реалізовано ядро серверу. Дане ядро являється веб-сервісом, типу REST. Для його реалізації використано Entity Framework з використанням підходу Code First, що дозволяє генерувати базу данних на основі класів-моделей написаних програмістом, далі для доступу до даних БД використаний паттерн Repository. Також використано паттерн UnitOfWork, який дозволяє спростит управління репозиторіями, яких є немало оскільки на кожну сутність-модель потрібен власний репозиторій. Для доступу до бази через інтернет використано концепцію ASP.NET MVC. Кожен контроллер керує власною таблицею, та містить різного роду атрибути для точної ідентифікації запитів з відповідними методами.;

- На основі вимог до системи, повинен бути адміністратор, який буде слідкувати за правильною роботу системи та відповідно взаємодіяти до його функціональних можливостей записаних у документації. Для нього реалізовано адміністративну частину, використовуючи технології WPF яка дозволяє будувати графічні додатки. Використано паттерн MVVM, який дозволяє відділити представлено від бізсне-логіки додатку;

- Згідно проаналізованих вимоги у другому розділі та спроектованого графічного інтерфейсу додатку здійснено реалізацію мобільної частини системи. Для цього використано швидко зростаючий у потужності та перспективах фреймворк Xamarin, який дозволяє писати на мові C# додатки, що можуть бути запущені на таких мобільних платформах, як: iOS, Android та WindowsPhone. Для цього додатку також було застосовано паттерн MVVM. Реалізовно збереження розкладу користувача використовуючи SQLite БД.

ВИСНОВКИ І ПРОРОЗИЦІЇ

У дипломному проекті «мобільна система управління навчальним процесом у ВНЗ» поставлена і завдання були наступпі: розробка алгоритму авторизації системи для мобільного клієнта, алгоритм реєстрації користувача, мобільний клієнт, серверну частину, адміністративну частину сервера.

Розроблено мобільний клієнт для система управління навчальним процесом, що може бути використаний для моніторингу навчання, як студентом так і викладачем. Дана система виконана задопомогою кросс-платформенного фреймворку тому може бути запущена на декількох ОС, хоча більше зорієнтована на ОС Android.

Розроблено серверну частину система, що складається з 2 модулів: адмін-панелі, для керуваннями данними сервера та власне ядра сервера, що може працювати незалежно від адмін-панелі. Для цього розроблено структуру бази данних та представлено її реалізацію.

Розроблено алгоритми авторизації з урахуванням особливостей мобільних систем та потреб до систему в цілому. Розроблено алгоритм реєстрації користувача на адмін-панелі, що також використовується про додаванні нового користувача.

Незважчаючи на те, що поставлені цілі завдання були досягнуті даний програмний продукт має можливості для додавання нового функціоналу та підвищення ефективності роботи. Для цього можна запропонувати наступні речі: покращити алгоритм реєстрації використовуючи двофакторну системи; покращити вибірку з бази данни додаючи більше фукнцій; покращити вибірку з бази данни використовуючи більш універсальних фукнцій; додати до мобільного клієнта локалізацію мови додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Електронний університет. Моніторинг успішності студентів. [Електронний ресурс]. - Режим доступу: http://eun.bmstu.ru/products/progress/.

2. КНЕУ. Журнал успішності студентів. [Електронний ресурс]. - Режим доступу: https://kneu.edu.ua/ua/Information_for/students/jurnal/;

3. АСУ. Автоматизована систем управління навчальним закладом «МКР». [Електронний ресурс]. - Режим доступу: http://mkr.org.ua/portalinfos/index/2/21;

4. АСУ. Портал АСУ навчальним процесом НЮУ імені Ярослава Мудрого. [Електронний ресурс]. - Режим доступу: http://acs.nlu.edu.ua/#;

5. Wikipedia. АСУ. [Електронний ресурс]. - Режим доступу: https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F;

6. Академік. Архітектура ПО. [Електронний ресурс]. - Режим доступу: http://dic.academic.ru/dic.nsf/ruwiki/146913;

7. Mongo-Software. Декстопні додатки. [Електронний ресурс]. - Режим доступу: http://mongo-software.ru/desktop;

8. Інтуіт. Обзор архітектур сучасних програмних систем. [Електронний ресурс]. - Режим доступу: http://www.intuit.ru/studies/courses/2314/614/lecture/13316;

9. Хабрахабр. Паттерни для новачків: MVC vs MVP vs MVVM. [Електронний ресурс]. - Режим доступу: https://habrahabr.ru/post/215605/;

10. MSDN. Архітектура програмного забезпечення. [Електронний ресурс]. -

Режим доступу: https://msdn.microsoft.com/ru-ru/hh144976.aspx;

11. Інтуіт. Відношенння на діаграмі використання. [Електронний ресурс]. - Режим доступу: http://www.intuit.ru/studies/courses/32/32/lecture/1004?page=2;

12. Rusuller. Концепція MVC для чайників. [Електронний ресурс]. - Режим доступу: http://ruseller.com/lessons.php?tag=179;

13. lurkmore. C Sharp. [Електронний ресурс]. - Режим доступу: http://lurkmore.to/C_Sharp;

14. CyberForum. Плюси та мінуси C#. [Електронний ресурс]. - Режим доступу: http://www.cyberforum.ru/csharp-net/thread442516.html;

15. Metanit. Введення в Entity Framework. [Електронний ресурс]. - Режим доступу http://metanit.com/sharp/entityframework/1.1.php;

16. Хабрахабр. Особливості розробки під Xamarin.Forms. [Електронний ресурс]. - Режим доступу: https://habrahabr.ru/company/devexpress/blog/263645/;

17. Хабрахабр. Детально про Xamarin. [Електронний ресурс]. - Режим доступу: https://habrahabr.ru/post/188130/;

18. Metanit. Xamarin та кросс-платформенна розробка. [Електронний ресурс]. - Режим доступу: http://metanit.com/sharp/xamarin/1.1.php;

19. Metanit. Перший додаток з Entity Framework. Підхід Code First. [Електронний ресурс]. – Режим доступу: http://metanit.com/sharp/mvc/1.1.php;

20. Metanit. Введення в ASP.NET MVC. [Електронний ресурс]. – Режим доступу: http://metanit.com/sharp/entityframework/1.2.php;

21. Xamarin. Sharing Code Options. [Електронний ресурс]. – Режим доступу: https://developer.xamarin.com/guides/cross-platform/application_fundamentals /building_cross_platform_applications/sharing_code_options/;

22. Xamarin. Xamarin Documentation. [Електронний ресурс]. – Режим доступу: https://www.xamarin.com/;

23. Хабрахабр. Досвід розробки на декількох проектах(Xamarin).

[Електронний ресурс]. – Режим доступу: https://habrahabr.ru/company/icl_services/blog/270051/]/;

24. Metanit. Введення в WPF. Особливості платформи. [Електронний ресурс]. – Режим доступу: http://metanit.com/sharp/wpf/1.php;

25. Professorweb. Механізм прив`язки данних. [Електронний ресурс]. – Режим доступу: https://professorweb.ru/my/WPF/binding_and_styles_WPF/level8/8_6.php

26. Metanit. Паттерн «Репозиторій» в ASP.NET. [Електронний ресурс]. – Режим доступу: http://metanit.com/sharp/articles/mvc/11.php;

27. Metanit. Паттерн «UnitofWork» в ASP.NET. [Електронний ресурс]. – Режим доступу: http://metanit.com/sharp/mvc5/23.3.php;