Науковий вісник НЛТУ України
Scientific Bulletin of UNFU
ISSN 1994-7836 (print)
ISSN 2519-2477 (online)
https://doi.org/10.36930/40360306
Article received: 30.03.2026
Article accepted: 29.05.2026
Article published: 29.06.2026
Correspondence author: N. M. Slobodian nazar.slobodian.pz.2022@lpnu.ua

UDC 004.[02+04+4+9]:786.2

***Н. М. Слободян, Л. М. Журавчак***
*Національний університет "Львівська політехніка" м. Львів, Україна*

# РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ ТРЕНУВАННЯ ГРИ НА ПІАНІНО

Проаналізовано причини обмеженого використання у сфері освіти, зокрема – музичній, генеративних моделей штучного інтелекту. Відзначено вищу точність та менший час надання зворотного відгуку протоколом MIDI (англ. *Musical Instrument Digital Interface*, універсальний цифровий стандарт, який дає змогу музичним інструментам, комп'ютерам і контролерам обмінюватися даними) порівняно з алгоритмами оброблення звуку, що поряд з поширеністю цифрових піаніно зумовило доцільність його використання у роботі. Встановлено, що наявні застосунки для навчання гри на піаніно передбачають вивчення композицій, виконання вправ на слух та перевірку музичної теорії, Однак, недостатньо акцентують увагу на практичній потребі гри на інструменті, майже не приділяють уваги ізольованим вправам на гру: читанню ритму або акордів. Для розроблення мобільного застосунку обрано платформу Android, яка забезпечує максимальну доступність для користувачів через поширеність пристроїв з цією операційною системою. Використано мову програмування Kotlin, офіційну та рекомендовану для створення застосунків під Android, яка водночас набуває популярності для розроблення застосунків, що працюють одразу на кількох операційних системах. За допомогою взаємодоповнювальних методологій Clean Architecture та Domain-Driven-Design створено застосунок, в якому бізнес-логіка є ізольованим ядром програми, захищеним від інфраструктури, фреймворків та баз даних. Розроблено моделі домену для генерування завдань різних типів з налаштовуваною складністю, перевіркою введення даних користувачем та надання зворотного відгуку. Імплементовано декілька алгоритмів оцінювання особливостей введення даних для різних режимів: точна гра потрібних нот, гра нот з можливим зсувом на кілька октав, гра в довільній тональності за умови збереження інтервальних відстаней між натисками. Використано стандарт шрифтів музичної нотації SMuFL для імплементації модуля відображення нотного стану під потреби вправ. Забезпечено інтеграцію розробленого застосунку з цифровими піаніно, які підтримують протокол MIDI через USB або Bluetooth, що підвищить ефективність засвоєння інформації користувачами-початківцями, покращить якість вивчення теорії та їхні навички гри на піаніно.

**Ключові слова:** музична освіта; Android-розробка; протокол MIDI; генерування музичних вправ; стандарт SMuFL; цифрове піаніно; проєктування програмного забезпечення.

## Вступ / Introduction

Розвиток цифрових мультимедійних і мережевих технологій та мобільних пристроїв істотно трансформував сферу освіти, зокрема – музичної. Традиційно навички гри на клавішних інструментах покращували відвідуванням музичної школи або індивідуальними заняттями із вчителем. Разом із засвоєнням музичної грамотності такий підхід давав змогу вивчати потрібну теорію, розвивати навички гри на клавішних інструментах та отримувати відгук від учителя. Його недоліком є те, що відвідування школи може бути незручним або неможливим для певних категорій осіб, адже воно залежить від багатьох факторів: економічне становище, доступність музичних закладів освіти в районі проживання тощо. Поява мережі Інтернет спростила доступ до навчальних ресурсів, дала змогу проводити онлайн-заняття. Особливо популярними стали засоби комп'ютеризованого навчання (ЗКН) на мобільних пристроях завдяки їхній широкій доступності та зручності особистого використання. Водночас, прогрес у технологіях мікроелектроніки та цифрового оброблення сигналів привів до появи сучасного покоління цифрових інструментів (цифрових піаніно та компактних MIDI-клавіатур), які стають дедалі популярнішими. З точки зору розроблення ЗКН істотною їхньою перевагою над акустичними інструментами є підтримка протоколу MIDI. Його застосування поєднує інтеграцію цифрових музичних інструментів з комп'ютерами, що дає змогу обробляти події з пристрою (гру користувача) у режимі реального часу з мінімальними затримками.

Актуальність цього дослідження пов'язана з постійною потребою навчання нових поколінь у сфері музики, зокрема – гри на піаніно. Відповідно, потрібні застосунки, які надаватимуть можливість розвивати як когнітивні (читання нот, розпізнання ключових музичних елементів на слух, розуміння іншої теорії), так і моторні навички у спосіб інтеграції навчального застосунку з цифровим піаніно, що забезпечить всебічний розвиток музичних вмінь учнів-початківців. Тому основним напрямом дослідження є аналіз потреб музикантів-початківців, розроблення зручного та ефективного зас-

**© Copyright 2026 by the author(s)**

**Слободян Назар Михайлович**, студент-бакалавр, кафедра програмного забезпечення.
**Email:** nazar.slobodian.pz.2022@lpnu.ua; https://orcid.org/0009-0005-0550-021X
**Журавчак Любов Михайлівна**, д-р техн. наук, професор, кафедра програмного забезпечення.
**Email:** liubov.m.zhuravchak@lpnu.ua; https://orcid.org/0000-0002-1444-5882

**Цитування за ДСТУ:** Слободян Н. М., Журавчак Л. М. Розроблення мобільного застосунку для тренування гри на піаніно. Науковий вісник НЛТУ України. 2026, т. 36, № 3. С. 62–71.
**Citation APA:** Slobodian, N. M., & Zhuravchak, L. M. (2026). Development of the mobile application for piano training. *Scientific Bulletin of UNFU*, 36(3), 62–71. https://doi.org/10.36930/40360306

тосунку для навчання гри на піаніно та покращення інших потрібних умінь.

*Об'єкт дослідження* – розроблення інтерактивного мобільного застосунку, призначеного для навчання гри на піаніно.

*Предмет дослідження* – методи та програмні засоби розроблення мобільного застосунку для тренування гри на піаніно, які дадуть змогу автоматизувати контроль правильності виконання завдань у режимі реального часу, забезпечити миттєвий зворотний зв'язок з наставником.

*Мета роботи* – розробити інтерактивний мобільний застосунок для тренування гри на піаніно засобами мови Kotlin, фреймворку Jetpack Compose й протоколу MIDI, а також патернів проєктування Clean Architecture і MVVM, який дасть змогу автоматизувати контроль правильності виконання завдань у режимі реального часу, забезпечуючи цим самим миттєвий зворотний зв'язок з мобільним наставником, та підвищити ефективність самостійного засвоєння музичної грамоти.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження:*

- проаналізувати схожі програмні рішення мобільних пристроїв та ключові потреби користувачів-початківців, що забезпечить формування функціональних і нефункціональних вимог до застосунку та дасть змогу визначити раціональні підходи до навчання гри на піаніно з його використанням;
- спроєктувати архітектуру мобільного застосунку на основі структури навчальної програми, що забезпечить подальше розроблення об'єктно-орієнтованих моделей для генерування музичних вправ;
- реалізувати модуль мобільного пристрою для оцінювання дій користувача, що дасть змогу в режимі реального часу порівнювати введені користувачем дані з вимогами навчального рівня та забезпечить надання зворотного зв'язку з мобільним наставником;
- розробити графічний інтерфейс користувача (GUI) та здійснити його інтеграцію з розробленим модулем мобільного пристрою з подальшим його тестуванням, що дасть можливість користувачу взаємодіяти з комп'ютером за допомогою візуальних елементів, тобто замість введення текстових команд у терміналі, користувач керує системою через вікна, кнопки, піктограми (іконки) та переліки меню.

**Аналіз останніх досліджень та публікацій.** У дослідженні [10] автор дослідив ефективність засобів комп'ютеризованого навчання. Експеримент провели для двох груп студентів. Перша група використовувала ЗКН у поєднанні зі звичними заняттями, а друга (контрольна) група брала участь у заняттях без ЗКН. Після 10 уроків проведено два тестування: теоретичне та практичне. У першому тесті студенти відповідали на перелік питань, а практичний тест потребував виконання композиції, яку оцінювали за різними категоріями (чіткість звучання, точність ритму, розташування пальців, координація рук тощо). Дослідження показало, що група, яка використовувала ЗКН, мала кращий показник запам'ятовування, ніж контрольна група. Утім значної різниці у вмінні гри на піаніно не виявили. Зважаючи на кращу ефективність ЗКН порівняно зі звичним підходом до навчання, автор рекомендував їхнє впровадження у закладах музичної освіти. Результат дослідження також показав, що ЗКН недостатньо залучають безпосередню взаємодію з інструментом.

У роботі [16] дослідили пристрій HoloLens 2 та застосунок на Unity, призначений для гри музичних творів з використанням звукового аналізу для виявлення зіграних нот та надання відгуку. Застосунок використовував доповнену реальність для відображення нот композиції: після калібрування камери з клавішами піаніно над ними з'являлась довга горизонтальна лінія. Пропонувалось три режими: перегляд (користувач спостерігав за рухом нот, які опускались згори до лінії з автоматичним відтворенням звуку), тренування (ноти біля лінії зупинялись на три секунди, впродовж яких користувач мав правильно їх зіграти), гра (ноти рухаються як у режимі перегляду, але користувач повинен вчасно натискати та відпускати клавіші). В останніх двох режимах для оцінювання гри на інструменті використовуються алгоритми звукового оброблення. Запропоновані композиції мали кілька режимів складності, починаючи із гри однією рукою поодиноких нот та закінчуючи синхронною грою обома руками складніших мелодій та послідовностей акордів. Після проведення низки експериментів щодо точності алгоритмів звукового оброблення та враження користувачів від застосунку дослідники зробили декілька висновків. По-перше, виявили затримки оброблення звукового сигналу, проблеми з діапазоном та точністю розпізнавання нот. По-друге, основними мотиваційними моментами для користувачів були новизна та цікавість до змішаної реальності, незважаючи на сприйняту корисність застосунку. Автори визнали, що хоч використання алгоритмів звукового оброблення має нижчу точність за MIDI, їхня актуальність для мотивації користувачів зберігається, але в теперішньому вигляді такий підхід швидше слугує для розваги та як допоміжний. Також автори навели способи до покращення алгоритмів звукового оброблення, які, можливо, наблизили б їх до MIDI щодо точності, зокрема, зауважили, що найкраще себе показав алгоритм швидкого перетворення Фур'є з вікном Блекмена-Гарріса з точністю 97,28 %.

У дослідженні [3] автори мали на меті провести аналіз 40 наукових робіт на тему використання моделей генеративного штучного інтелекту для персоналізації та генерування вмісту для музичного навчання. Утім, зі 40 робіт тільки 13 стосувались генерування навчального вмісту, з них 8 – музичного. Під час аналізу виявили, що хоч зацікавленість у використанні генеративного штучного інтелекту та обчислювальних можливостей загалом існує, вона переважно стосується музичної теорії, а не інструментальної практики. Прикладами таких використань (з різними ступенями успішності) є створення акомпанементу для основних вправ, аналіз композицій задля виявлення потенційних проблемних місць, відгук про імпровізацію музиканта, створення персоналізованих навчальних планів. Також автори висловили сподівання, що генеративні моделі зможуть бути краще використані в контексті навчання музиці та для сприяння людській креативності, на противагу створення музики корпораціями з комерційними цілями.

У роботі [6] досліджено питання оцінювання гри на музичному інструменті в реальному часі за допомогою введення MIDI-даних. Розроблено застосунок для комп'ютерів на операційній системі Linux, який оцінював вчасність натискань та правильність зіграних нот послідовності, забезпечуючи зворотний зв'язок з мобільним наставником. Автор обґрунтував потребу створення таких рішень тим, що отримання зворотного зв'язку від викладача можливе тільки впродовж обмеженого часу

(наприклад, раз на тиждень на занятті). У роботі розглянуто три основні параметри MIDI-повідомлення (висоту ноти, силу натискання та час) та застосовано їх для реалізації алгоритму оцінювання дій користувача. Використаний автором музичний апарат залишається актуальним.

У науковій праці [4] дослідники проаналізували вплив технологій мобільного навчання на результати гри на фортепіано в онлайн-освіті. Вони довели, що використання мобільних застосунків із підтримкою гейміфікації та інтерактивних інструментів для практики значно підвищує мотивацію та ефективність самостійного засвоєння матеріалу порівняно з традиційними методами. Водночас, у дослідженні наголошено на адаптації інтерфейсу до потреб користувачів, що свідчить про критичну важливість розроблення застосунків у сфері музичної освіти.

У дослідженні [12] автори проаналізували впровадження алгоритмів машинного навчання в систему музичної освіти для оцінювання виконання композиції та надання зворотного зв'язку з мобільним наставником у режимі реального часу. Вони довели, що автоматизоване оброблення звукових сигналів та аналіз таких елементів, як висота тону, ритм і динаміка, дають змогу формувати дієві підказки. Водночас, використання складних предикативних моделей для оброблення звуку потребує значних обчислювальних ресурсів, що підкреслює актуальність легших підходів (наприклад, на основі MIDI) для гарантування мінімальних затримок відгуку та енергоефективності застосунку.

Отже, аналіз останніх досліджень та публікацій засвідчує потребу у детальному описі програмної архітектури сучасних мобільних застосунків для музичної освіти. Відомі рішення страждають від затримок оброблення звуку [12, 16], не мають інтеграції з музичним інструментом [10] або орієнтовані на застарілі платформи [6]. Потрібно заповнити цю прогалину, розробивши збалансоване рішення для виконання сфокусованих тренувальних вправ. Відтак, існує потреба в детальному обґрунтуванні архітектури мобільного застосунку, яка поєднує об'єктно-орієнтоване генерування завдань із миттєвим оцінюванням MIDI-даних.

**Матеріали та методи дослідження.** Під час аналізу методів вирішення поставлених завдань та способів їхньої практичної реалізації розглянуто великі комерційні застосунки для вивчення пісень та їх користувацьку документацію (SimplyPiano [18], Yousician [24], Flowkey [1]), орієнтовані на навчання з акустичними піаніно і часткову підтримку MIDI-інструментів, та спеціалізовані тренажери (Perfect Ear, musictheory.net), у яких подано ефективні вправи на основні теми музичної теорії та різні типи слухових навичок.

***Обрані технології та обґрунтування.*** Через специфіку призначення застосунку (можливість використання в поєднанні з інструментом) платформа має бути портативною та невеликою. Як платформу обрано мобільні пристрої, бо персональні комп'ютери не можна розмістити на пюпітрі. Оскільки однією з цілей розроблення застосунку є збільшення доступності до музичної освіти, то Android є привабливішою платформою (серед мобільних пристроїв 67,35 % використовують Android, а iOS – тільки 32,55 % [14]). Інструменти для розроблення на Android є безкоштовними, натомість розроблення для iOS можлива тільки за наявності комп'ютера на macOS і смартфона на iOS [17]. Використання багатоплатформних рішень не має сенсу, для проведення інтерактивних вправ важливими є мінімізація затримок й вища продуктивність, які характерні нативним застосункам, окрім того, вони займають менше місця на диску, хоча їхнє розроблення може бути складнішим [15].

Обрано мову програмування Kotlin, бо вона є офіційною та рекомендованою для створення застосунків під Android. Для роботи з локальною базою даних, MIDI-пристроями та для впровадження залежностей використано бібліотеки від Android: Room, Android MIDI, Hilt відповідно [2]. Для створення графічного інтерфейсу користувача обрано фреймворк Jetpack Compose для Kotlin [22]. Він призначений для декларативного розроблення графічного інтерфейсу та роботи з підходами MVVM (*Model-View-ViewModel*) та UDF (*Unidirectional Data Flow*). Це дає змогу повністю синхронізувати життєвий цикл графічних компонентів з бізнес-логікою.

Використано офіційне для Android середовище розробки Android Studio [13], яке надає інструменти для розроблення, тестування, відлагодження та оптимізації релізів застосунку.

Для зручного, гнучкого та швидкого відображення музичних елементів та нотного стану використано стандарт шрифтів музичної нотації SMuFL. Векторна природа шрифтів гарантує масштабованість без втрати якості та забезпечує стилістичну узгодженість нотації.

## Результати дослідження та їх обговорення / Research results and their discussion

***Загальна архітектура мобільного застосунку.*** З огляду на те, що основна логіка застосунку, окрім такого функціоналу, як виконання транзакцій для купівлі рівнів (який використовує інфраструктуру Google), працює безпосередньо на пристрої користувача, обрано архітектуру товстого клієнта. Увесь вміст також зберігається локально, тому використання тонких клієнтів та інших мережевих архітектур не має сенсу, оскільки тільки збільшило б тривалість затримки відгуку застосунку та потребувало б з'єднання з інтернетом.

Для організації коду використано принципи чистої архітектури, за якої система повинна мати кілька шарів, залежності між якими йдуть тільки всередину [19]. Цього можна досягнути, відокремивши логіку домену (моделі, бізнес-логіка та інтерфейси, які не мають жодних залежностей, окрім самої мови Kotlin та інших класів домену), інфраструктури (імплементації інтерфейсів домену та інший код, який має залежності від зовнішніх бібліотек та сервісів, зокрема – для збереження даних) та презентації (код для створення графічного інтерфейсу, включно з Jetpack Compose, моделі вигляду).

Розмежування повноважень у контексті взаємодії інтерфейсу користувача та моделі досягається використанням MVVM та UDF. Модель вигляду (*ViewModel*) діє як посередниця, яка перетворює та робить спостережуваним стан моделі. Вигляд є пасивним та тільки відображає те, що йому надано. При цьому, згідно з UDF, дані рухаються тільки вгору (від моделі до вигляду), а події – вниз (від вигляду до моделі), проходячи через модель вигляду.

Сукупно така архітектура мобільного застосунку відповідає принципам розділення відповідальності, забезпечує надійність (мінімальна залежність від мережі та сторонніх сервісів майже повністю унеможливлює

проблеми з серверами), тестованість (модулі програми логічно розділені, домен та презентацію можна тестувати незалежно), модифікованість (частини легко можна змінити або замінити), автономність, супроводжуваність, безпеку (особисті дані користувача та дані про покупки обробляються та зберігаються сервісами Google).

Основні частини застосунку та залежності між ними відображено на рис. 1. На цьому рисунку зображено такі елементи: Piano Training App – розроблюваний застосунок; Domain – шар, що інкапсулює основну логіку застосунку (теорію музики, алгоритми для генерування завдань для вправ та оцінювання дій користувача, потрібні інтерфейси для інверсії залежності тощо); Music Theory Core – основні сутності моделі, необхідні для подання музичних даних та роботи з ними; Pedagogy Engine відповідає за проведення вправ (генерування запитань, оброблення та оцінювання особливостей введення даних користувачем); AudioEngine – інтерфейси для роботи зі звуком (метроном, відтворення мелодій тощо); Curriculum Manager містить інтерфейси для отримання доступних рівнів, збереження прогресу та моделі для подання модулів, вправ та рівнів; Presentation відповідає за відображення даних та оброблення дій; View Models перетворює дані з домену для відображення, передає події в домен для оброблення; Jetpack Compose UI відображає дані з моделей відображень, делегує їм події користувача; Data відповідає за збереження даних, роботу зі зовнішніми системами та бібліотеками, імплементує інтерфейси домену; Audio Player імплементує інтерфейси домену для роботи зі звуком; Preferences (DataStore) використовує DataStore для збереження налаштувань застосунку; Relational Storage (Room) відповідає за збереження прогресу користувача з використанням бібліотеки Room; Billing Adapter – компонент для оброблення платежів задля визначення доступу користувача до деяких частин застосунку; MIDI Adapter перетворює вхідні MIDI повідомлення інструментів у стандартизовану форму домену; Google Play Services – зовнішній сервіс для автентифікації, авторизації, виконання платежів та перевірок покупок.

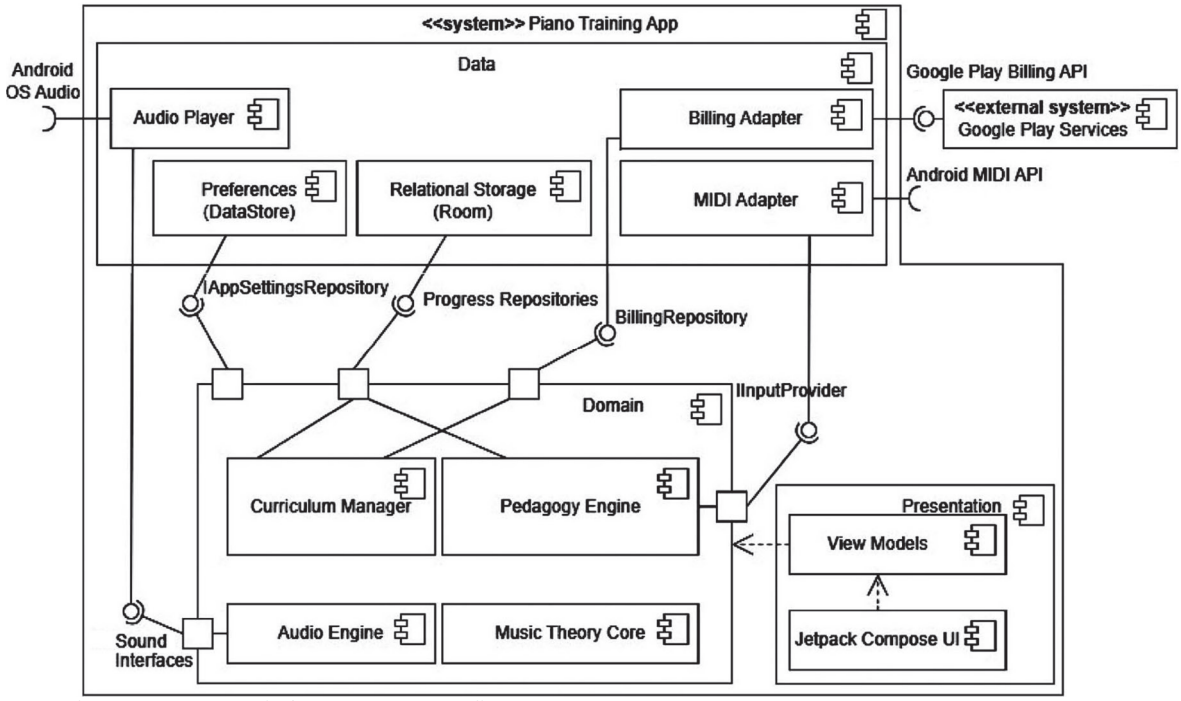


**Рис. 1.** UML-діаграма компонентів / UML component diagram

На рис. 2 зображено основні артефакти розгортання застосунку: Android Smartphone – пристрій користувача, на якому виконується програма та зберігаються дані; PianoRise.apk – скомпільована програма; app_database.db – файл збереження прогресу користувача бібліотекою Room; user_prefs.pb – файл збереження налаштувань застосунку бібліотекою DataStore; Google Play Servers – сервери Google Play, які використовуються для оброблення покупок; Digital Piano / MIDI Keyboard – пристрій, здатний надсилати MIDI повідомлення; Audio Output – пристрій відтворення звуку.

UML-діаграма розгортання (англ. *Deployment Diagram*) – це схема, яка показує фізичне обладнання та пристрої, на яких працює комп'ютерна програма. Вона допомагає зрозуміти, куди саме встановлюється код застосунку та як різні комп'ютери з'єднуються між собою. Водночас, розгортання застосунку – це процес перенесення готової програми на сервер або в магазин додатків, щоб нею могли користуватися інші люди. Головна мета цього етапу – зробити так, щоб програма працювала швидко, без помилок і була постійно доступною для користувачів.

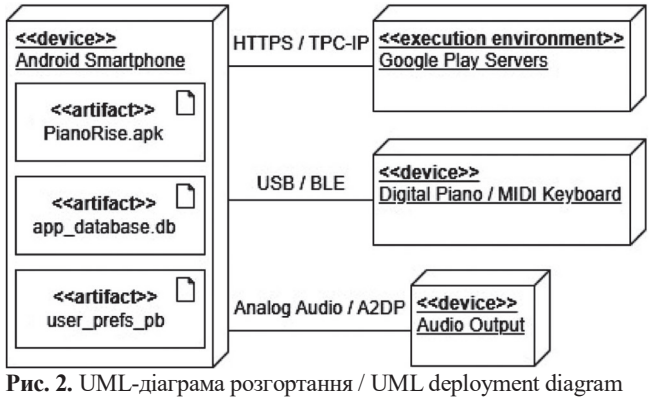


**Рис. 2.** UML-діаграма розгортання / UML deployment diagram

***Подання нот та основних елементів.*** Більшість вправ застосунку оперують з такими базовими елементами музики: інтервали, акорди та гами. Їх типи подано як переліки IntervalType, ChordType та ScaleType, які

імплементують інтерфейс IntervalStructure. На рис. 3 зображено діаграму класів для цих елементів.

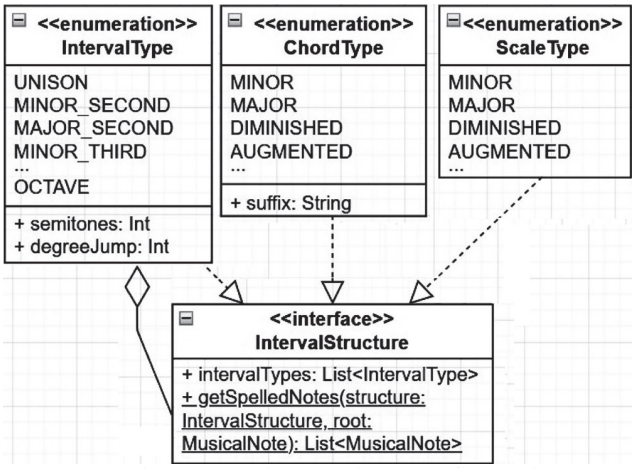


**Рис. 3.** UML-діаграма класів IntervalStructure / IntervalStructure UML class diagram

Оскільки усі інші елементи будуються з інтервалів, важливо правильно їх подати. З точки зору створення звуку, має значення тільки розмір інтервалу у півтонах. Але для відображення нот цього недостатньо. Згідно з MIDI, відлік номерів нот ведеться з 0, де 12 – нота $C_0$, а 60 – $C_4$. Припустимо, що потрібно знайти ноту, яка є на малу терцію вище, ніж 60. Оскільки розмір малої терції в півтонах дорівнює 3, отримаємо ноту 63 (чорна клавіша між $D_4$ та $E_4$), що правильно. Однак, за відображення виникне питання: "Який знак альтерації потрібно використати?" Цю ноту можна позначити як $D♯_4$, $E♭_4$, $F♭♭_4$ тощо. Тому для однозначного визначення знаку альтерації також потрібно зберігати розмір інтервалу у кроках, який означає відстань у нотах. Для малої терції крок дорівнює 2, тобто, якщо починати з $C_4$ (неважливо з яким знаком альтерації), потрібно пройти дві ноти вгору ($D_4$ та $E_4$). Знаючи, що цільова нота повинна бути $E_4$ (MIDI-номер 64), а фактично маємо ноту з номером 63, знаходимо різницю між фактичною та очікуваною висотами (–1). Оскільки фактична нота на один тон нижча за $E_4$, потрібно застосувати знак альтерації ♭. Такий підхід гарантує правильний підбір знаку альтерації, зважаючи на інтервальний контекст.

З огляду на ці правила, клас подання висоти ноти повинен зберігати не тільки висоту звуку, але й свою назву та знак альтерації. Назва зберігається як кількість кроків від $C_0$ (у такому разі $C_4$ матиме позицію 28). Будь-яке з цих значень можна обчислити, знаючи інші два, але оскільки ці дані потрібні для відображення та відтворення звуку, доцільно зберегти їх під час створення.

Формули для обчислення MIDI-номера з позиції та знаку альтерації, знаку альтерації з MIDI-номера та кроку, кроку з MIDI-номера та знаку альтерації запишемо так:

$$M = M_{base} + 12\left\lfloor \frac{S}{7} \right\rfloor + D(S \bmod 7) + A,$$

$$A = M - (M_{base} + 12\left\lfloor \frac{S}{7} \right\rfloor + D(S \bmod 7)),$$

$$S = 7\left\lfloor \frac{\Delta}{12} \right\rfloor + D^{-1}(\Delta \bmod 12),$$

де: $D = M – A – M_{base}$; $S$ – крок від $C_0$ ($C_0 = 0$); $M$ – абсолютний MIDI- номер; $A$ – значення знаку альтерації у півтонах ($\mu = -1$, $o = 1$, $\nu = 0$); $M_{base}$ – MIDI значення для $C_0$ (12); $D(c) \in \{0, 2, 4, 5, 7, 9, 11\}$ – функція зміщення, яка визначає кількість півтонів під базової ноти $C$ для заданого ступеня $c \in \{0, 1, 2, 3, 4, 5, 6\}$ (де 0 – це $C$, 1 – $D$ і т. ін.); $D^{-1}(k)$ – обернена функція, яка повертає клас ступеня за відстанню у півтонах (тільки для білих клавіш). Запис $S \bmod 7$ обчислює клас ступеня ноти незалежно від її октави (повертає значення від 0 до 6), що дає змогу функції $D$ визначити відповідне зміщення у півтонах. Вираз $\Delta \bmod 12$ обчислює відстань у півтонах у межах октави, що дає змогу оберненій функції $D^{-1}$ визначити клас ступеня базової ноти.

Класи для подання висоти ноти подано на рис. 4.

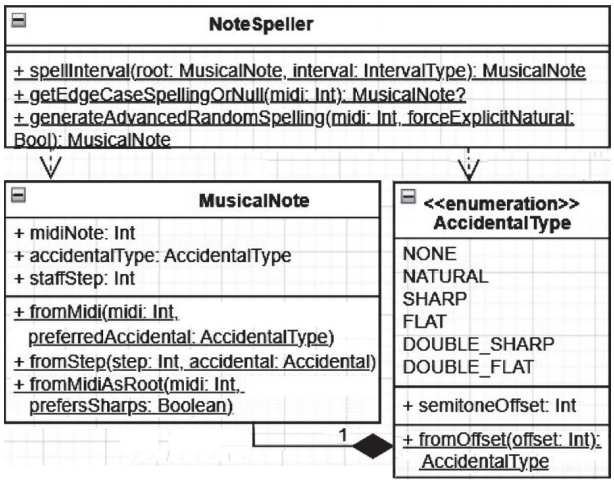


**Рис. 4.** UML-діаграма класів подання висоти ноти / UML class diagram of classes representing notes' height

Окрім висоти, ноти мають і часові характеристики. З музичної точки зору, ноти мають основні тривалості (повна, половинна, четвертна тощо), до яких можна застосувати модифікатори (подовження на половину). На основі цієї інформації під час відображення обираються тип голови ноти, наявність штиля, тип прапорця тощо. Також потрібно зберігати відносну тривалість ноти. Для цього використовується стандартне для багатьох систем значення TPQN (поділів на четвертні ноти), яке становить 480. Використовуючи коєфіцієнти тривалостей щодо четвертної, можна знайти інші значення (наприклад, для восьмої 480*0,5 = 240). Знаючи розмір ноти в поділах та кількість ударів на хвилину, можна обчислити позицію та тривалість ноти в мілісекундах. На рис. 5 зображено класи, пов'язані з часовими властивостями нот.

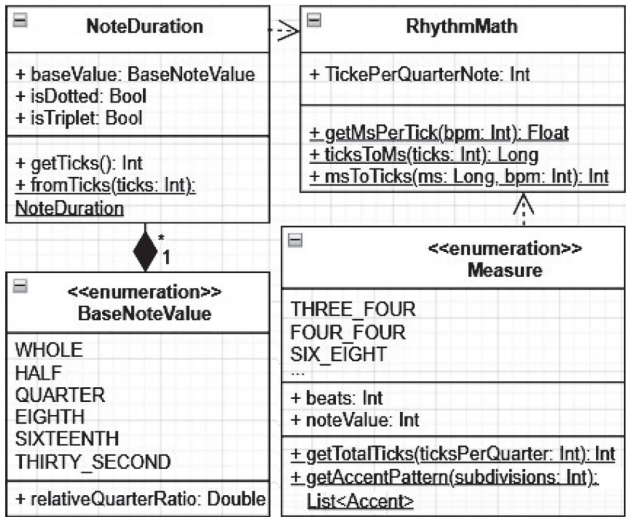


**Рис. 5.** UML-діаграма класів, пов'язаних з часовими властивостями нот / UML class diagram of classes, related to notes' temporal properties

Наведені вище моделі використовуються для подання музичних елементів (нот та пауз), ієрархію яких зображено на рис. 6. UML-діаграма класів для основних музичних елементів будується навколо головного класу Музичний Твір (Composition), який об'єднує в собі Доріжки (Track), Ноти (Note), Паузи (Rest) та Акорди (Chord). Ця схема показує, з яких частин складається пісня та як ці частини пов'язані між собою.

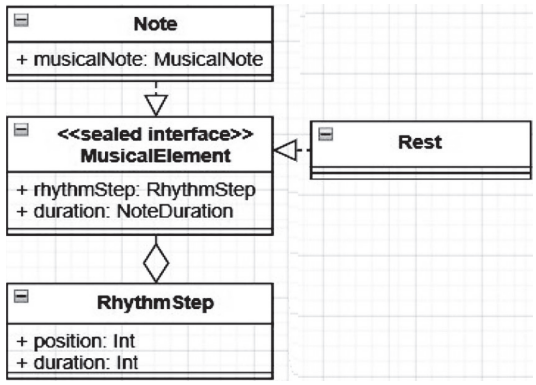


**Рис. 6.** UML-діаграма класів основних музичних елементів / UML class diagram of main musical elements

Для подання запитання використовується ієрархія класів MusicalConcept, які містять усі дані, потрібні для отримання переліку нот. Наприклад, інтервал містить початкову ноту, тип акорду, режим відтворення (гармонічний, мелодійний вгору або вниз, тривалість нот). Ритм містить перелік музичних елементів (нот та пауз), тактовий розмір, кількість тактів (рис. 7). UML-діаграма класів музичних концептів відображає структуру музичної системи, де головними сутностями є Виконавець (Artist), Альбом (Album), Трек (Track) та Плейлист (Playlist). Ця схема показує, як музичні елементи пов'язані між собою у програмах на кшталт Spotify чи Apple Music.

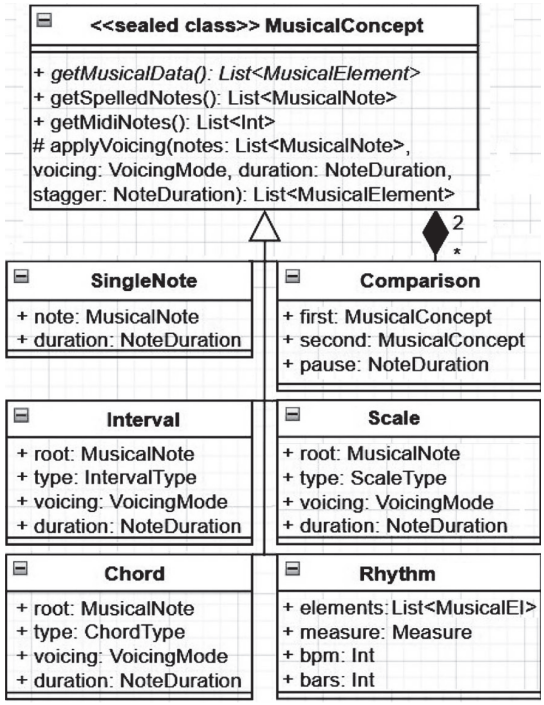


**Рис. 7.** UML-діаграма класів музичних концептів / UML class diagram of musical concepts

***Подання запитань та їхнє генерування.*** UML-діаграма класів для системи музичних вправ складається з головного класу вправи, базового класу для запитань та кількох окремих класів для різних типів завдань (наприклад, розпізнавання нот чи інтервалів). Для створення запитання потрібно кілька додаткових класів, зображених на рис. 8. Перший з них – конфігурація відображення. Вона містить такі параметри: потреба відображення нотного стану, назва музичного концепту, бажаний ключ та параметри відтворення (дозвіл на автоматичне відтворення концепту, відтворення користувачем) та інші. Другий клас – вимоги до введення даних. Створено три варіанти: вікторина, на час та піаніно. Такий підхід дає змогу комбінувати різні концепти, вимоги відображення та введення даних користувачем для створення багатьох типів вправ. Прикладами типових вправ є вікторина на вибір типу відтворюваного акорду, вибір більшого із двох інтервалів, гра акорду з його назви, гра ноти з нотного стану, гра ритму, відображеного на нотному стані. Однак, є й недолік – деякі комбінації не мають семантичного сенсу (наприклад, ввід користувачем ритму в режимі одночасного оцінювання натисків клавіш). Вирішення цієї проблеми наведено далі.

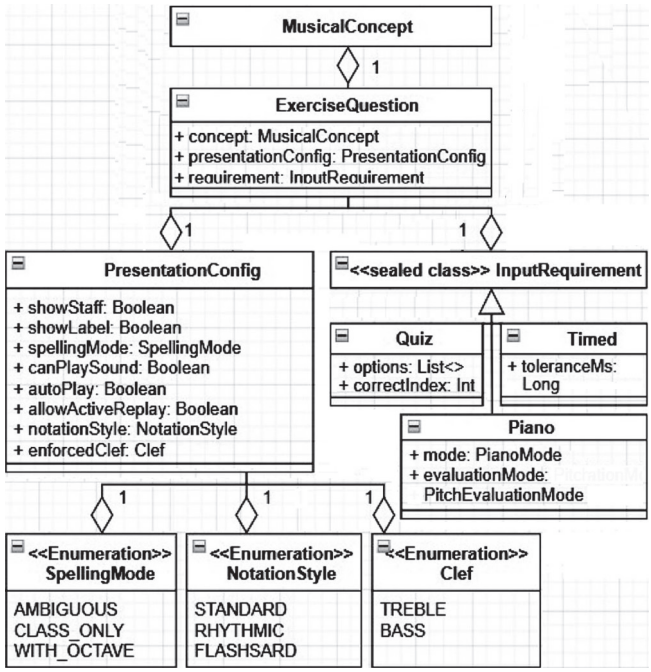


**Рис. 8.** UML-діаграма класів запитань музичних вправ / UML class diagram for music exercise questions

Створення запитань використовується з допомогою набору генераторів. На рис. 9 зображено імплементовані генератори та їхні параметри. Особливої уваги заслуговують параметри voicingStrategy (гармонічна, мелодійна, випадкова), requirementBuilder (використовується для створення об'єкта InputRequirement для створеного концепту). UML-діаграма класів для генератора запитань показує структуру системи, яка створює тести або опитування з тексту чи бази даних. Для створення системи, яка генерує запитання для музичних вправ (наприклад, на вгадування нот, інтервалів чи акордів), найкраще використати об'єктно-орієнтовану структуру.

***Збереження доступних типів завдань та рівнів.*** Застосунок має зберігати перелік доступних типів вправ і рівнів. Зберігати усі рівні безпосередньо як об'єкти-генератори незручно хоча б тому, що потрібно постійно дублювати функцію генерування вимоги до введення даних користувачем для кожного рівня. Для вирішення цієї проблеми створено класи-креслення для вправ. Вони зберігають потрібні дані для створення ге-

нератора, а внутрішня логіка забезпечує створення правильного об'єкта вимоги до введення даних. Створення генераторів музичних вправ виконується з допомогою об'єкта-фабрики. Кожен рівень разом з об'єктом-кресленням зберігає ідентифікатор та кількість запитань. На вищому рівні вміст зберігається у модулях, які містять ідентифікатор, перелік тем теорії та типів вправ. Кожен тип вправи містить перелік рівнів. У програмуванні для створення складних об'єктів (таких як різні типи генераторів музичних вправ) часто використовують шаблон проєктування "Фабрика" (англ. *Factory Object* або *Factory Method*).

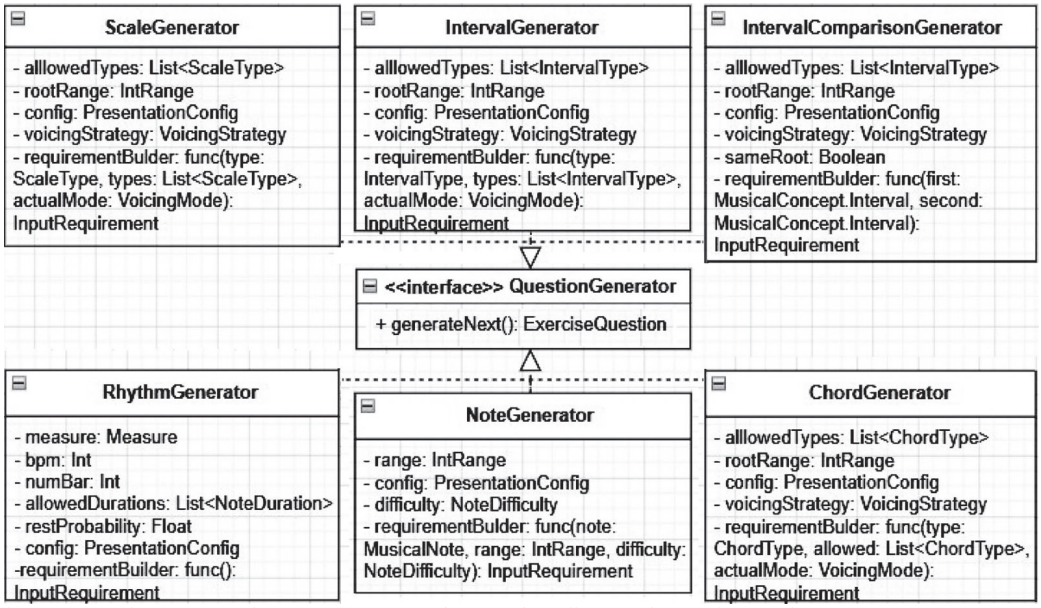


**Рис. 9.** UML-діаграма класів генераторів музичних вправ / UML class diagram for music exercise generators

***Перевірка введення даних та зворотний відгук.*** Усі дії, які може зробити користувач в межах вправи, перетворюються в доменні об'єкти. Такими діями можуть бути вибір варіанта відповіді, натиск на клавішу піаніно, натиск на кнопку ритму. Їх перелік та вміст зображено на рис. 10.

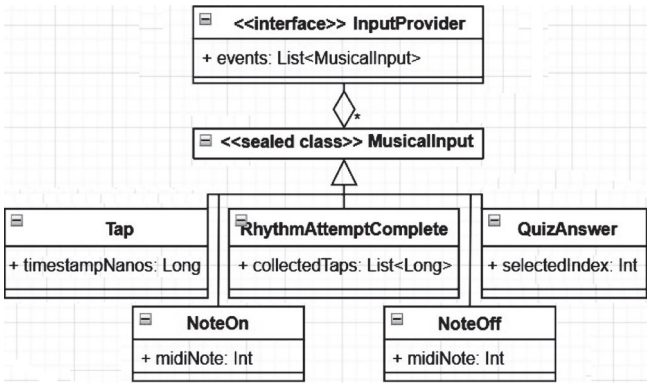


**Рис. 10.** UML-діаграма класів подій введення даних / UML class diagram of input events

Усі класи оцінювання особливостей введення даних працюють за одним принципом: на вхід отримують MusicalInput та ExerciseQuestion, а повертають один з результатів (Correct, Progress, Incorrect, Waiting, RhythmReview, SequentialProgress, SequentialComplete). Нижче подано опис двох створених алгоритмів оцінювання. Узагальнену послідовність подій під час виконання типового завдання подано на рис. 11. Роль Coordinator полягає в координації деяких інших частин (метроном та засоби відтворення звуку, об'єкт сесії для буферизації подій тощо). ComprehensiveExercise відповідає за відстеження прогресу в межах рівня, делегації оцінювання особливостей введення даних, нарахування балів, завершення рівня. UML-діаграма послідовностей для оброблення вхідної події показує, як різні частини програми спілкуються між собою у часі після того, як трапилося якась подія (наприклад, користувач натиснув якусь кнопку).

*Жадібний алгоритм оцінювання процедури відтворення ритму*:

Крок 1. Отримуємо перелік натисків з часовими мітками, де 0 мс – початок ритму.

Крок 2. З ExerciseQuestion отримуємо перелік нот та пауз ритму, перетворюємо їхні часові характеристики, збережені у поділах, у мілісекунди з допомогою RhythmMath та даних про кількість ударів на хвилину (див. рис. 5).

Крок 3. Для кожної ноти ритму знаходимо найближчий натиск та його різницю від ноти у мілісекундах. Якщо різниця менша за розмір ідеального вікна, позначаємо як "Perfect", якщо менша за розмір вікна толерантності, то – "Early" або "Late", залежно від знаку відхилення.

Крок 4. Якщо натиск виходить за межі вікон або є зайвим, позначаємо як "Miss".

Крок 5. Формуємо об'єкт RhythmReview, який містить перелік відгуків, кожен з яких зберігає час відгуку й мітку. Також об'єкт має прапорець, який позначає наявність хоча б однієї мітки "Miss".

За високого темпу вікна толерантності можуть перекриватись. Знайшовши спершу найближчі пари нот та натисків, забезпечуємо вибір найточніших збігів. Паузи під час підбору пар при цьому ігноруються. Алгоритм карає хаотичні натиски.

*Алгоритм висотного оцінювання музичних вправ у режимі довільної октави*:

Крок 1. Отримуємо довільний перелік натиснутих клавіш (MIDI-номери).

Крок 2. З ExerciseQuestion отримуємо перелік нот (еталон).

Крок 3. Сортуємо обидва масиви за зростанням MIDI-номерів.

Крок 4. Якщо різниця між найменшим та найбільшим номерами входу та аналогічна різниця еталону не є однаковими, повертаємо "Incorrect".

Крок 5. Кожну натиснуту ноту перевіряємо за модулем 12 з еталоном. Якщо є хоча б одна зайва нота, повертаємо "Incorrect".

Крок 6. Обчислюємо базове зміщення (різницю між найнижчими нотами натисків та еталону). Перевіряємо, чи зберігається це зміщення для усіх інших пар нот та чи кратне воно 12. Якщо ці умови не дотримані, повертаємо "Incorrect". Якщо умови дотримані, але натиснутих нот менше, ніж в еталоні, повертаємо "Progress". Якщо умови дотримані та кількість нот однакова, повертаємо "Correct".

Цей алгоритм корисний для вправ на гру акордів за типом та початковою нотою або для гри гам, оскільки дає можливість гри у різних октавах для тренування гнучкості гравця.

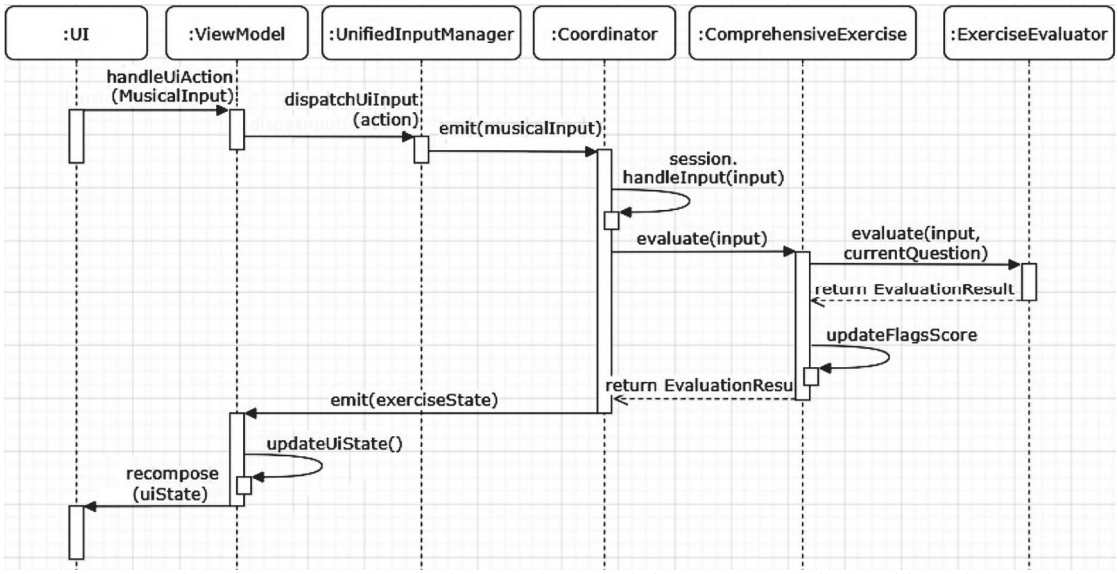


**Рис. 11.** UML-діаграма послідовностей типового оброблення вхідної події / UML sequence diagram illustrating a typical process for handling an incoming event

***Реалізація користувацького інтерфейсу та тестування застосунку.*** Створено графічний користувацький інтерфейс потрібних сторінок. Для гнучкості застосунку розроблено модуль для рендерингу нот, який обробляє елементи поточного ExerciseQuestion та інший контекст (натиснуті ноти) для створення переліку відображених об'єктів. Обчислюються позиції елементів з урахуванням режиму відображення та доступного на екрані місця. З використанням стандарту SMuFL малюється нотоносець та інші музичні символи. Взаємодія починається з головного меню. З нього можна перейти на сторінку під'єднання до пристрою, налаштувань, вибору вправ та рівнів. Виконано тестування інтеграції застосунку із двома музичними інструментами: цифровим піаніно Roland FP-30x та MIDI-модулем застосунку FL Studio у режимі Piano Roll з використанням засобів зв'язку Bluetooth та USB відповідно.

Отже, розроблений музичний навчальний програмний застосунок дасть змогу учням музичних закладів відпрацьовувати потрібні їм когнітивно-моторні навички, є кориснішим у контексті музичної освіти внаслідок проведення вправ з нотним станом, мобільним і зручнішим для використання за піаніно, доступним засобом для музикантів-аматорів.

Розвиток нового функціоналу може відбуватися у таких напрямах: проєктування та створення досконаліших ЗКН, використання штучного інтелекту для генерування та оцінювання вправ, його поєднання з доповненою реальністю для надання зворотного відгуку та відображення нот.

**Обговорення результатів дослідження.** Розроблення мобільного застосунку для тренування гри на піаніно – це перспективний, але технічно складний проєкт, успіх реалізації якого залежить від точного розпізнавання нот та гейміфікації процесу навчання. Щоб створити конкурентоспроможний продукт рівня Simply Piano або Yousician, потрібно чітко розпланувати архітектуру, вибрати технологічний стек та реалізувати ключовий функціонал.

У праці [25] наведено модель електронного навчання, яка поєднує технології віртуальної реальності та глибокого машинного навчання (TASTFT, STGNN, RL-CGAN) для моніторингу точності ритму, висоти звуку та жестів, демонструючи високі показники точності (96,86 % для гри на фортепіано). Незважаючи на високу точність оброблення мультимодальних сенсорних даних, подібні VR-системи потребують великих обчислювальних потужностей та дорогої інфраструктури.

У дослідженні [23] розглянуто модель інтелектуального навчання з використанням цифрових аудіотехнологій та алгоритмів машинного навчання, інтегровану у роботехнічну систему для дошкільної музичної освіти. Підхід довів свою ефективність у підвищенні мотивації учнів та автоматизації підбору навчальних матеріалів. Експериментальні результати підтверджують, що такі системи ефективно допомагають у вивченні музичної теорії та розвитку навичок гри.

Автори публікації [7] використали логарифмічні фільтри у частотній області, зокрема двошарову ReLU, для видобування частот з музичного сигналу. Сконструйовану згорткову нейронну мережу з константним Q-перетворенням використано для формування матриць гармонічних послідовностей. Упровадження дворівневих стратегій класифікації дає змогу підвищити загальну точність розпізнавання сигналів для багатьох інструментів (для фортепіано показник F1 досягає 0,936).

У праці [11] автори проаналізували особливості використання методів оброблення природної мови (NLP) для символьного генерування музики. Зазначено, що завдання NLP та пошуку музичної інформації (MIR) є схожими, що спричинило адаптацію методів NLP для оброблення символьних музичних даних. Незважаючи на перспективність таких моделей для генерування музики, проблемними завданнями є токенізація музичних

послідовностей, пов'язаних у відмінностях звичайного тексту та музики.

Автори роботи [21] розробили систему "PianoBand", призначену для використання як портативну альтернативу піаніно. До неї входить наручний пристрій, що інтегрує інерційний вимірювальний модуль (IMU), мініатюрна RGB-камера під зап'ястям та роздрукований аркуш клавіатури з маркерами для надійного відображення клавіш на плоских поверхнях. Завдяки легкому конвеєру оброблення даних IMU та машинного зору в реальному часі забезпечуються високоточна взаємодія, підтримка гри поодиноких нот, акордів, динамічна швидкість натискання та базові прийоми артикуляції. Перевірка системи показала високу відносну точність виявлення дотиків (понад 0,99).

У дослідженні [5] розглянуто проблему наявних моделей генерування звуку щодо забезпечення точного створення мелодій, ритмічних малюнків. Для подолання цих обмежень запропоновано модуль "Loops Transformer", який використовує багатокроковий механізм перехресної уваги для інтеграції текстових умов та вхідного MIDI-сигналу. Додаткове впровадження інструментально-орієнтованого навчання з підкріпленням дало змогу авторам досягти покращення об'єктивних метрик якості звуку (зниження FAD на 1,3) та підвищити точність слідування мелодії на 12 % за метрикою Dynamic Time Warping.

Автори публікації [9] представили рішення проблеми подання одноканальних MIDI-треків як партитур для лівої та правої рук. Вони застосували алгоритм, побудований на основі довгої короткочасної пам'яті та методів зіставлення для оброблення неструктурованого файлу. Цей підхід дає змогу зручно подати обидві партитури на нотному стані, що покращить досвід користувачів для гри з нот.

У праці [8] запропоновано мультимодальну нейромережеву структуру, яка об'єднує Mel-спектрограми, низькорівневі дескриптори та символьні MIDI-дані через механізми крос-модальної уваги. З її допомогою можна виявляти емоційне забарвлення музичних творів. Точність моделі перевершила наявні підходи, тому підтвердила факт, що символьні MIDI-інструкції несуть унікальну та невіддільну інформацію про музичну експресію.

У роботі [20] автори розглянули задачу перетворення зображень нотного стану в MIDI-файл за умов обмеженого часу та обчислювальних ресурсів мобільного пристрою. Для досягнення високої швидкодії використано алгоритми комп'ютерного зору для виявлення висоти та позиції нот з їхнього вигляду, позиції відносно ліній нотного стану на основі правил західної музичної нотації. Фінальне вирівнювання виконується з допомогою динамічного програмування. Вдалось досягти високої точності (F measure дорівнює 0,89) з середньою тривалістю обчислень 0,9 с.

Отже, внаслідок обговорення результатів дослідження виявлено, що більшість нових рішень орієнтовано на важкі моделі штучного інтелекту [5, 7, 8, 25], що потребують значних обчислювальних ресурсів, або на складні специфічні апаратні комплекси [21, 23], які є малодоступними. Питання детермінованої автоматизації розглядають тільки в розрізі постоброблення чи багатомодального пошуку [9, 11, 20]. Жодне з проаналізованих досліджень не пропонує мобільного рішення з описом внутрішньої архітектури, здатного процедурно генерувати вправи на пристрої, відображати ноти з використанням стандарту SMuFL, виконувати оцінювання гри у інших октавах або тональностях.

Відтак, розроблений підхід успішно заповнює науково-технічну прогалину, пропонуючи автономну та енергоефективну архітектуру програмного застосунку, який передбачає якісну взаємодію з користувачем у реальному часі.

Отже, внаслідок виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

*Наукова новизна отриманих результатів дослідження* – удосконалено модель автоматизованого оцінювання дій користувача під час гри на фортепіано, яка, на відміну від відомих рішень, забезпечує інваріантність перевірки музичного виконання щодо транспозиції в інші октави чи ключі, що дає змогу автоматизувати контроль правильності виконання завдань у режимі реального часу, забезпечити миттєвий зворотний зв'язок з мобільним наставником.

*Практична значущість результатів дослідження* – розроблений програмний застосунок для Android можна використовувати для покращення когнітивних та моторних навичок майбутнього музиканта, пов'язаних з грою на цифрових піаніно.

## Висновки / Conclusions

Розроблено інтерактивний мобільний застосунок для тренування гри на піаніно засобами мови Kotlin, фреймворку Jetpack Compose й протоколу MIDI, а також патернів проєктування Clean Architecture і MVVM, який дає можливість автоматизувати контроль правильності виконання завдань у режимі реального часу, забезпечуючи цим самим миттєвий зворотний зв'язок з мобільним наставником, та підвищити ефективність самостійного засвоєння музичної грамоти. За результатами проведеного дослідження можна зробити такі висновки.

1. Розглянуто деякі тенденції розвитку застосунків музичної освіти, підходи до залучення користувачів, дослідження ефективності таких застосунків. Проаналізовано популярні мобільні застосунки та сервіси для вивчення музичної теорії та гри на піаніно. Обрано підхід, який має сприяти виконанню ізольованих вправ на слух та на гру за інструментом, що тренуватиме когнітивні та моторні навички користувача.
2. Обрано інструменти для створення нативного, продуктивного та доступного застосунку на Android. Інтегровано під'єднання цифрових інструментів, які підтримують протокол MIDI для проведення практичних вправ на гру з мінімальною затримкою відгуку застосунку та максимальною точністю розпізнавання нот. До таких вправ належать розпізнавання елементів на слух або за нотним поданням, гра елементів з нотного стану, назви або звучання, гра ритму, поданого як ноти та інші.
3. Спроєктовано та імплементовано необхідні модулі для генерування завдань, перевірки даних, введених користувачем, і надання зворотного відгуку та інших допоміжних функцій (збереження переліку рівнів вправ, метроном тощо).
4. Розроблено модуль відображення нотного стану з використанням шрифту Bravura стандарту SMuFL, який забезпечив стилістично злагоджений рендеринг музичної нотації потенційному користувачу.
5. Тестування застосунку з цифровим піаніно Roland FP-30X підтвердило його здатність залучати у грі когнітивні й моторні навички для виконання вправ та зручність його використання.

## References

1. Advice and answers from the flowkey team. *flowkey*. URL: https://help.flowkey.com/en/ (date of ACCESS: 12.05.2026).
2. Androids kotlin-first approach. *Android Developers*. URL: https://developer.android.com/kotlin/first
3. Carnovalini, F., Espírito Santo, L., & Wiggins, G. A. (2025). Personalized music education: A systematic review of AI generation methods. *IEEE ACCESS, 13*, 207433–207447. https://doi.org/10.1109/ACCESS.2025.3639898
4. Chen, B., & Tang, R. (2025). Factors influencing mobile learning technology adoption and learning outcomes in online piano education: A comparative study of Chinese and Russian students. *PLoS ONE*, *20*(8), article ID e0330269. https://doi.org/10.1371/journal.pone.0330269
5. Chen, G. Y., & Soo, V. W. (2024). Controllable music loops generation with MIDI and text via multi-stage cross attention and instrument-aware reinforcement learning. *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 6851–6859. https://doi.org/10.1145/3664647.3681187
6. Duvall, D. (2008). Real-time MIDI Performance Evaluation for Beginning Piano Students. All Theses, article ID 438. URL: https://open.clemson.edu/all_theses/438
7. Gao, X., & Gao, R. (2025). Music signal recognition aids based on convolutional neural networks in music education. *Systems and Soft Computing*, 7, article ID 200219. https://doi.org/10.1016/j.sasc.2025.200219
8. Huang, Z., Wu, M., Shao, X., Xu, M., & Wang, C. (2025). Cross-level feature fusion for music emotion recognition: Integrating mel-spectrogram, high-level musical descriptors and symbolic MIDI data with attention mechanisms. *2025 4th International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology* (*AIoTC*), pp. 124–134. https://doi.org/10.1109/AIoTC66747.2025.11198797
9. Jin, Y., & Wang, M. (2020). LSTM model for single to dual track piano MIDI file. *2020 IEEE 9th Global Conference on Consumer Electronics* (*GCCE*), pp. 29–31. https://doi.org/10.1109/GCCE50665.2020.9291967
10. Kaleli, Y. S. (2020). The effect of computer-assisted instruction on piano education: An experimental study with pre-service music teachers. *International Journal of Technology in Education and Science* (IJTES), *4*(3), 235–246. URL: https://www.researchgate.net/publication/342446243
11. Le, D. V. T., Bigo, L., Herremans, D., & Keller, M. (2025). Natural language processing methods for symbolic music generation and information retrieval: A survey. *ACM Computing Surveys*, *57*(7), article ID 175. https://doi.org/10.1145/3714457
12. Lu, L. (2025). AI-powered intelligent music education systems for real-time feedback and performance assessment. *International Journal of Information and Communication Technology*, *26*(18), 33–47. https://doi.org/10.1504/IJICT.2025.146690
13. Meet android studio. *Android Developers*. URL: https://developer.android.com/studio/intro (date of ACCESS: 12.05.2026)
14. Mobile operating system market share worldwide. *StatCounter Global Stats*. URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (date of ACCESS: 12.05.2026)
15. Panchal, D. N. (2026). Evaluating Cross-Platform Vs Native Mobile Application Development: A Practical Case Study. *International Journal of Creative and Open Research in Engineering and Management*, *02*(04). https://doi.org/10.55041/ijcope.v2i4.456
16. Putranda, I. G., Yumna, A. M., Rosmansyah, Y., & Sukmana, Y. (2023). Exploring audio processing in mixed reality to boost motivation in piano learning. *IEEE ACCESS*, *11*, 71194–71200. https://doi.org/10.1109/ACCESS.2023.3293250
17. Running macOS and Xcode on a Windows PC: what actually works and what doesnt. *Medium*. URL: https://medium.com/@bsrkayaoglukurt/running-macos-and-xcode-on-a-windows-pc-what-actually-works-and-what-doesnt-5991d4d3912f
18. Simply Piano. *Simply Piano help center*. URL: https://pianohelp.hellosimply.com/en/collections/3968889-simply-piano
19. The Clean Architecture. *Clean Coder Blog*. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html
20. Tsai, T. J., Yang, D., Shan, M., Tanprasert, T., & Jenrungrot, T. (2020). Using cell phone pictures of sheet music to retrieve MIDI passages. *IEEE Transactions on Multimedia*, *22*(12), 3115–3127. https://doi.org/10.1109/TMM.2020.2973831
21. Wang, Z., Li, Z., Li, C., Ao, Z., Feng, J., & Zhou, J. (2026). PianoBand: A multimodal wristband interface for portable piano interaction. *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems*, article ID 1085, 1–21. https://doi.org/10.1145/3772318.3790607
22. Why compose. *Android Developers*. URL: https://developer.android.com/develop/ui/compose/why-adopt
23. Xu, Y. (2024). Intelligent e-learning system in the development of preschool music education based on digital audio technology. *Entertainment Computing*, 50, article ID 100682. https://doi.org/10.1016/j.entcom.2024.100682
24. Yousician support. *Yousician.* URL: https://support.yousician.com/hc/en-us/categories/360003836718
25. Zhao, K. (2026). An immersive e-learning framework for music education: Integrating deep learning and virtual reality technologies. *Expert Systems with Applications*, 305, article ID 130881. https://doi.org/10.1016/j.eswa.2025.130881



***N. M. Slobodian, L. M. Zhuravchak***

*Lviv Polytechnic National University, Lviv, Ukraine*


## DEVELOPMENT OF THE MOBILE APPLICATION FOR PIANO TRAINING


The reasons for the limited application of generative artificial intelligence models in education, particularly in music one, are analyzed. The higher accuracy and lower feedback latency of the MIDI (Musical Instrument Digital Interface) protocol – a universal digital standard enabling data exchange between musical instruments, computers, and controllers – compared to audio processing algorithms are highlighted. Along with the widespread adoption of digital pianos, this justified the feasibility of utilizing the MIDI standard in this work. It has been established that existing piano-learning applications focus on song learning, ear-training exercises, and music theory testing, yet underemphasize the practical necessity of actual playing, dedicating minimal attention to isolated performance exercises such as rhythm reading or chord execution. The Android platform was selected for the mobile application development to ensure maximum user ACCESSibility due to the market prevalence of devices running this operating system. The Kotlin programming language – the official and recommended choice for Android development, which is also gaining traction for cross-platform application development – was utilized. By leveraging the complementary methodologies of Clean Architecture and Domain-Driven Design, the application was engineered where the business logic forms an isolated application core, shielded from infrastructure, frameworks, and databases. Domain models were developed to handle the generation of various task types with customizable complexity, input validation, and real-time feedback provision. Several input evaluation algorithms were implemented for distinct operational modes: precise performance of required notes, note execution with a permissible octave shift, and playing in an arbitrary key provided that the relative musical intervals between keystrokes are preserved. The SMuFL (Standard Music Font Layout) standard was adopted to implement a musical staff rendering module tailored to the requirements of the exercises. Seamless integration with digital pianos supporting the MIDI protocol via USB or Bluetooth was achieved, thereby enhancing the learning efficiency for novice users, improving music theory retention, and advancing their practical piano-playing skills.



**Keywords:** music education; Android development; MIDI protocol; music exercise generation; SMuFL standard; digital piano; software design.