МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,

освітньо-професійної програми «Інформаційні технології проектування»

на тему: «Програмний додаток створення та редагування гітарних табулатур»

Здобувача (ки) групи ІТ-03 Маховика Олександра Сергійовича
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к.т.н., старший викладач Едуард КУЗНЄЦОВ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«__» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Маховика Олександра Сергійовича

1 Тема роботи *Програмний додаток створення та редагування гітарних табулатур*

керівник роботи *Кузнєцов Едуард Генадійович, к.т.н., старший викладач* _____,

затверджені наказом по університету від «*24*» травня 2024 р. №00579-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи Технічне завдання на розробку програмного додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вступ, аналіз предметної області, моделювання програмного продукту, розробка програмного продукту, висновки

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Актуальність, постановка задачі, аналіз комплексів-аналогів, порівняння аналогів, структура проекту, вимоги до продукту, структурно-функціональна модель, варіанти використання програмного додатку, варіанти використання програмного додатку, засоби реалізації, демонстрація програмного додатку, висновки

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання 01.01.2024

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області	01.01.2024-23.01.2024	Виконано
2	Планування робіт	23.01.2024-12.02.2024	Виконано
3	Моделювання програмного продукту	12.02.2024-01.03.2024	Виконано
4	Розробка програмного додатку	01.03.2024-28.04.2024	Виконано
5	Тестування програмного додатку	28.04.2024-16.05.2024	Виконано
6	Створення документації проекту	16.05.2024-26.05.2024	Виконано

Студент _____ Олександр Маховик
(підпис)

Керівник роботи _____ к.т.н., Едуард КУЗНЄЦОВ
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Програмний додаток створення та редагування гітарних табулатур».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 14 найменувань, додатків. Загальний обсяг роботи − 70 сторінок, у тому числі 42 сторінок основного тексту, 1 сторінка списку використаних джерел, 27 сторінок додатків.

Метою роботи є дослідження та аналіз програмних аналогів, формування технічного завдання, моделювання програмного додатку, розробка програмного додатку а також демонстрація роботи програмного додатку.

У першому розділі проведено ретельний аналіз предметної області та сформовано постановку задачі та розроблено технічне завдання проекту.

Другий розділ представляє собою дослідження структурно функціональних діаграм та діаграми варіантів використання та їх створення.

У розділі 3 детально описується розробка програмного додатку та повноцінний опис методів програмного коду.

Результатом проведеної роботи є програмний додаток створення та редагування гітарних табулатур, який допоможе новачкам зрозуміти основи гітарних табулатур.

Практичне значення полягає у створенні гітарної табулатури, можливість її редагування, музичного відтворення та збереження у файлову систему.

Ключові слова: програмний додаток, табулатура, Java, JavaFX, Java Sound API,файл, гітара, тестування.

ЗМІСТ

ВСТУП

У музичному світі наразі велика увага зосереджується на розвитку програмного забезпечення, спрямованого на музикантів та композиторів. Однією з галузей, що знаходиться у центрі уваги, є створення та редагування гітарних табулатур - популярний метод запису музичних композицій для гітари. У зв'язку з цим, основним завданням моєї дипломної роботи є розробка програмного додатку, який надає можливість користувачам створювати та редагувати гітарні табулатури за допомогою сучасних технологій.

Користь цієї розробки полягає в спрощенні та полегшенні процесу створення музики для гітаристів на будь-якому рівні кваліфікації. Використання програмного додатку для створення гітарних табулатур дозволить користувачам з легкістю записувати свої музичні ідеї, вносити зміни та редагувати композиції за допомогою зручного інтерфейсу. Такий додаток має велике значення в сучасному музичному середовищі, де все більше музикантів шукають ефективні інструменти для творчості та виразності.

Головною метою моєї дослідницької роботи є створення програмного забезпечення, яке дозволить користувачам легко створювати та редагувати гітарні табулатури, розширюючи їх можливості у написанні музики та виконанні на гітарі, а також ділитися створеними табулатурами з іншими користувачами.

Для досягнення мети мого проекту необхідно виконати наступні завдання:

- Аналіз потреб користувачів: Спрямувати увагу на оцінку актуальності створення програмного додатку для створення та редагування гітарних табулатур. Важливо дослідити, які проблеми стикаються музиканти при використанні існуючих програмних рішень та які потреби вони мають у зручному інструменті для творчості.
- Огляд існуючих рішень: Провести докладний аналіз існуючих програмних продуктів, які спеціалізуються на створенні та редагуванні музики. У

контексті гітарних табулатур, важливо з'ясувати, які функції мають існуючі програми, їхні переваги та недоліки, щоб врахувати це при розробці власного програмного забезпечення.

- Проектування інтерфейсу та функціоналу: Розробити концепцію програмного додатку для створення та редагування гітарних табулатур. Це включає визначення основних функцій програми, таких як створення табулатур, редагування, можливість збереження та експорту файлів у різні формати.
- Реалізація програмного продукту: На основі розробленої концепції, розробити та втілити програмний додаток. Це включає написання коду, інтеграцію різних функцій та тестування для забезпечення правильної роботи програми.
- Тестування та вдосконалення: Провести тестування розробленого програмного додатку для перевірки його функціональності та зручності використання.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень та публікацій

На сучасному етапі розвитку інформаційних технологій індустрія музичного програмного забезпечення переживає значний ріст. Однак, серед цього розмаїття додатків, спрямованих на потреби музикантів та композиторів, відсутній належно розвинений інструмент для гітаристів у формі програми для роботи з гітарними табулатурами. Ця проблема стає особливо актуальною, оскільки гітаристи, особливо початківці, постійно шукають зручний інструмент, який дозволить їм ефективно працювати з табулатурами.

Незважаючи на те, що існують деякі програми, призначені для створення та редагування гітарних табулатур, багато з них мають обмеження та недоліки. Деякі програми можуть бути надто складними для новачків, не надаючи необхідного рівня інтуїтивності та зручності у використанні. Інші можуть бути недостатньо функціональними для задоволення потреб більш досвідчених користувачів, які вимагають розширених можливостей для редагування та аранжування музики у форматі табулатур. Крім того, деякі програми обмежують можливості обміну табулатурами між користувачами або експортування їх у різні формати файлів.

Наявність недоліків у функціональності та доступності існуючих програм для роботи з гітарними табулатурами підкреслює необхідність подальших досліджень і розробки програмного забезпечення, яке б забезпечувало зручність, ефективність та доступність для всіх категорій користувачів, що працюють з гітарними табулатурами.

Отже, наявність недоліків у функціоналість та доступності існуючих програм для роботи з гітарними табулатурами підкреслює необхідність подальших досліджень і розробки програмного забезпечення, яке б забезпечувало зручність,

ефективність та доступність для всіх категорій користувачів, що працюють з гітарними табулатурами.

1.2 Аналіз існуючих продуктів – аналогів

Оцінка аналогів програмного забезпечення для створення та редагування гітарних табулатур дозволяє виявити різноманітність наявних рішень на ринку та їхні особливості. Переглянемо три ключові програми: Songsterr, Tux Guitar та Guitar Pro.

Songsterr славиться своєю простотою та доступністю. Цей продукт пропонує онлайн-платформу для пошуку, перегляду та відтворення гітарних табулатур без необхідності встановлення додаткового програмного забезпечення. Його інтерфейс є досить інтуїтивно зрозумілим, але функціонал обмежений, часом відсутні деякі продвинуті можливості, такі як редагування або створення табулатур та імпорт/експорт у файл.

Guitar Pro є однією з найпопулярніших програм у цій галузі. Вона славиться своїм розширеним функціоналом, включаючи можливість створення, редагування та відтворення гітарних табулатур, а також велику базу додаткових можливостей, таких як імпорт та експорт MIDI-файлів. Проте інтерфейс Guitar Pro може виглядати перенавантаженим та не дуже зручним для новачків.

Tux Guitar, натомість, є відкритою та безкоштовною альтернативою Guitar Pro. Ця програма має більший функціонал у порівнянні з Songsterr, зокрема, можливість редагування табулатур, налаштування звучання та експорт файлів у різні формати. Однак інтерфейс Tux Guitar може виглядати менш інтуїтивно для новачків та користувачів, які звикли до іншого типу програм.

У підсумку, порівнюючи ці програми, можна зазначити, що кожна з них має свої переваги та недоліки. Songsterr вражає простотою та доступністю, Tux Guitar -

безкоштовністю та розширеним функціоналом, а Guitar Pro - високою функціональністю та популярністю.

Наведемо детальне порівняння кожної з програм.

«Guitar Pro»

- Розширені можливості створення та редагування табулатур та нот: Програма дозволяє точно створювати табулатури для гітари та інших інструментів. Інтерфейс програми інтуїтивно зрозумілий, що полегшує введення та редагування нот, акордів і ритмічних малюнків.
- Відтворення звуку для перевірки композиції: Guitar Pro забезпечує відтворення створених табулатур за допомогою високоякісних звукових бібліотек, що дозволяє музикантам прослуховувати свої композиції у реальному часі та вносити необхідні зміни.
- Підтримка багатьох інструментів: Програма підтримує не тільки гітару, але й інші інструменти, такі як бас-гітара, ударні, клавішні та інші, що робить її універсальним інструментом для створення повноцінних аранжувань.
- Режим вивчення для навчання гри на гітарі: Включає інтерактивні уроки та вправи, які допомагають користувачам освоювати техніку гри на гітарі, з можливістю грати разом з табулатурами.
- Доступність на різних платформах: Guitar Pro доступний для Windows, macOS, Android та iOS, що дозволяє користувачам працювати з програмою на різних пристроях.
- Платний з пробним періодом: Хоча програма платна, вона зазвичай пропонує пробний період, що дозволяє оцінити її функціональність перед придбанням.

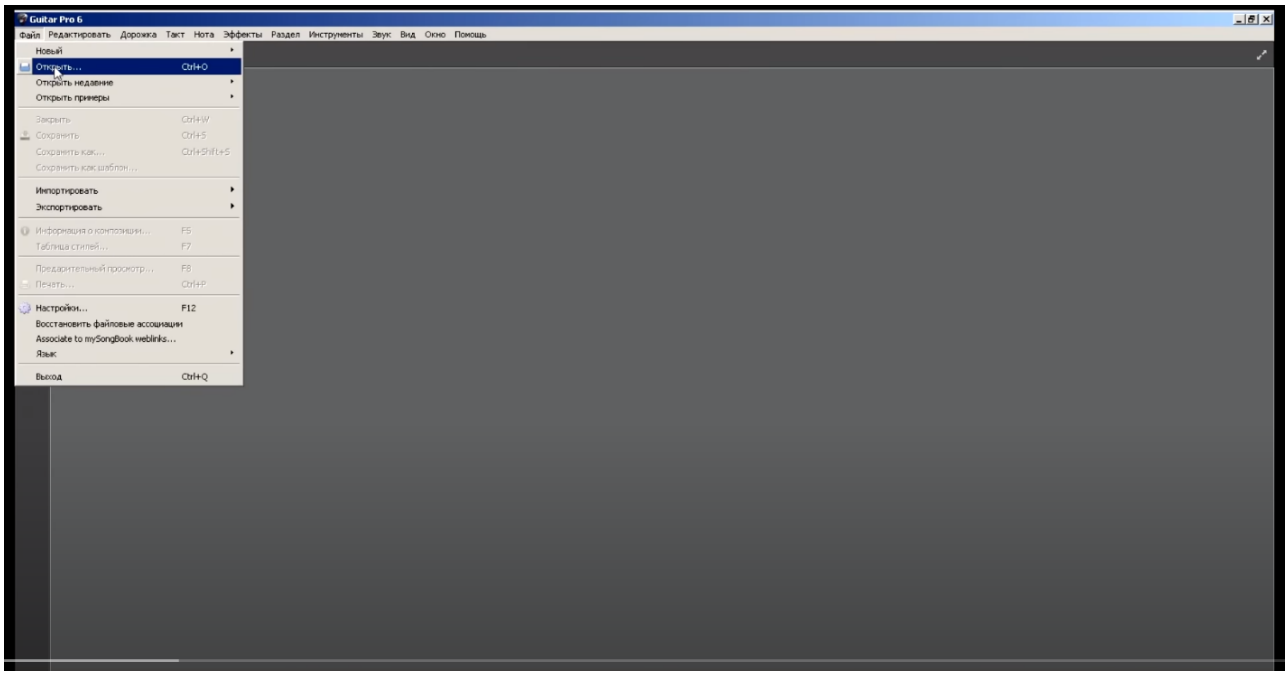

Рисунок 1.1 – Головна сторінка

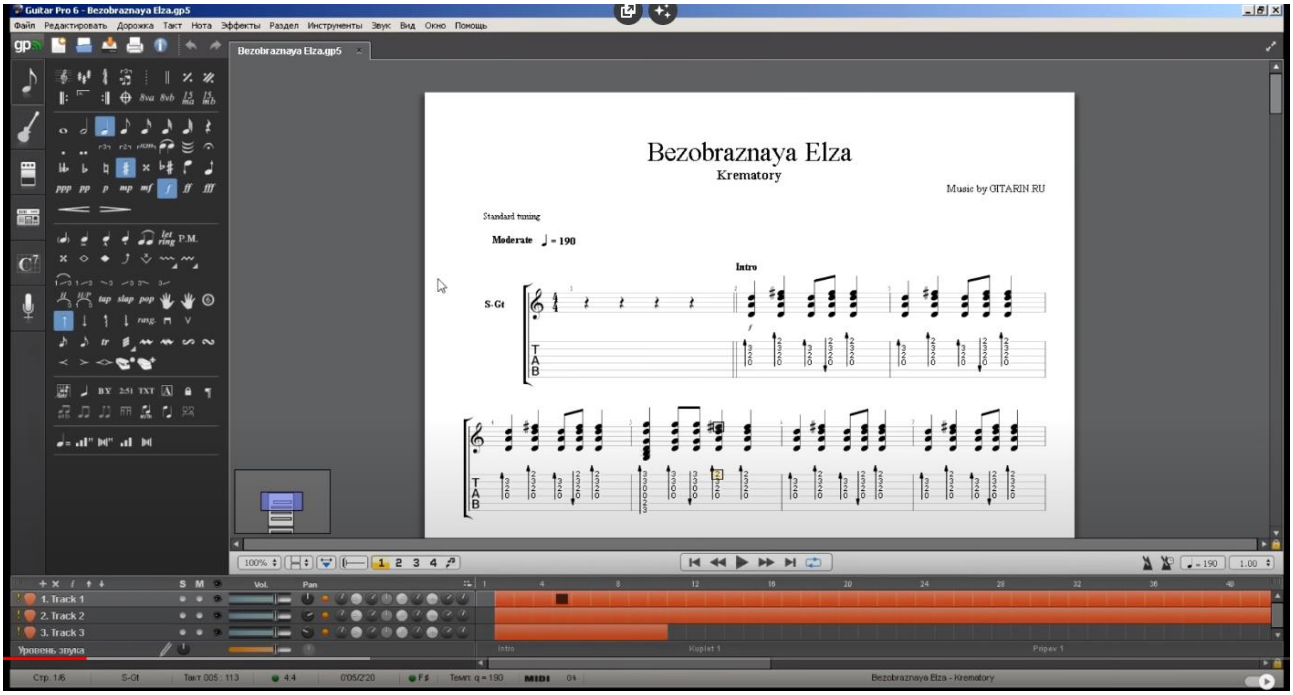

Рисунок 1.2 – Головний інтерфейс

«Songsterr»

- Велика онлайн-база табулатур та нот.

- Відтворення звуку для вірного відтворення композицій.
- Веб-сервіс та мобільні додатки.
- Веб-сервіс та додатки для Android та iOS.
- Безкоштовний, але можлива підписка для додаткових функцій.

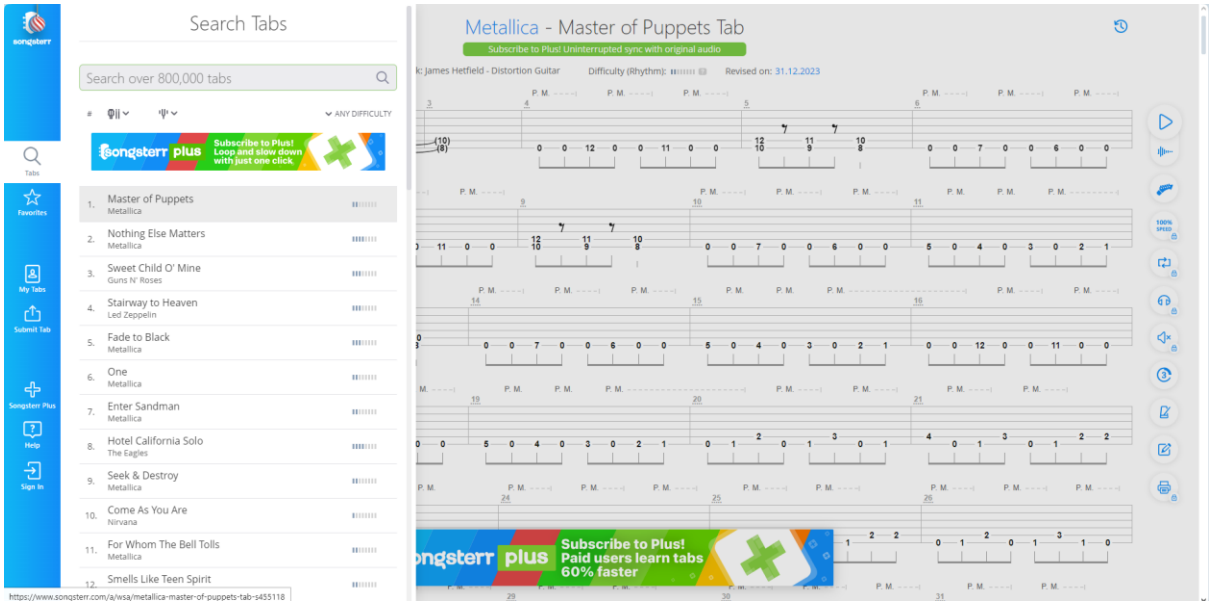

Рисунок 1.3 – Головна сторінка “ Songsterr ”

«TuxGuitas»

- Безкоштовний та відкритий.
- Базовий функціонал для створення та редагування табулатур.
- Доступний для Windows, macOS та Linux.

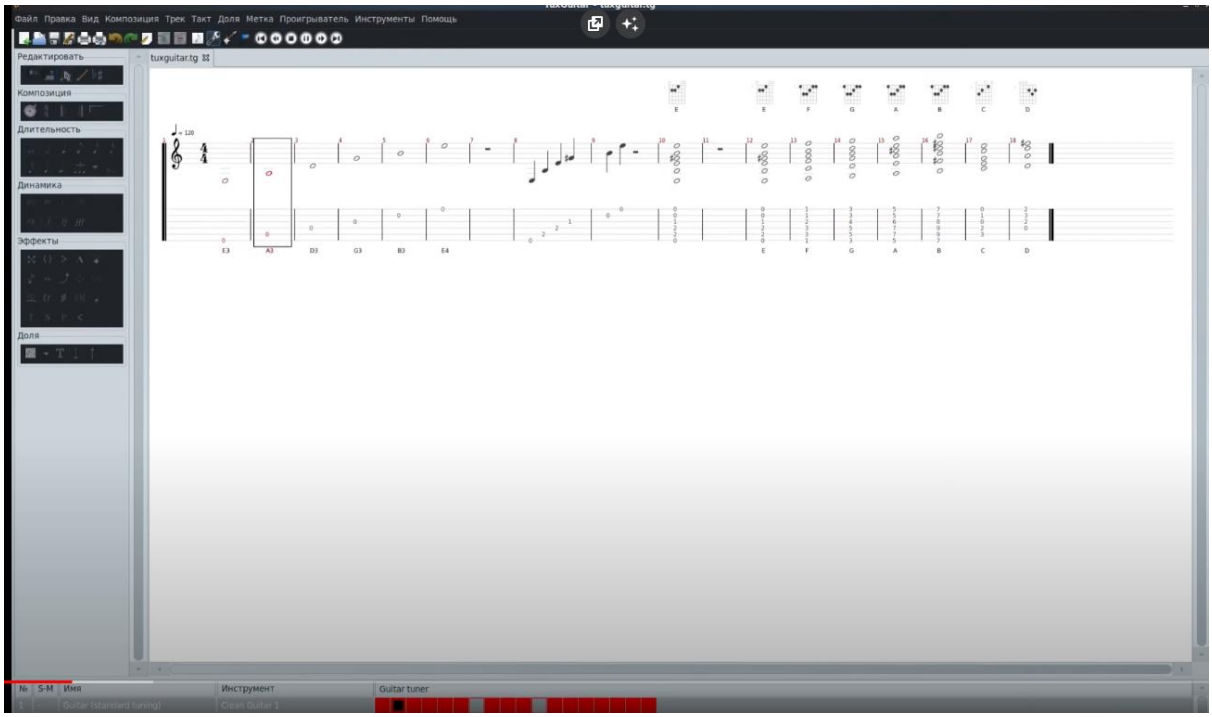

Рисунок 1.4– Головна сторінка “ TexGuitar”

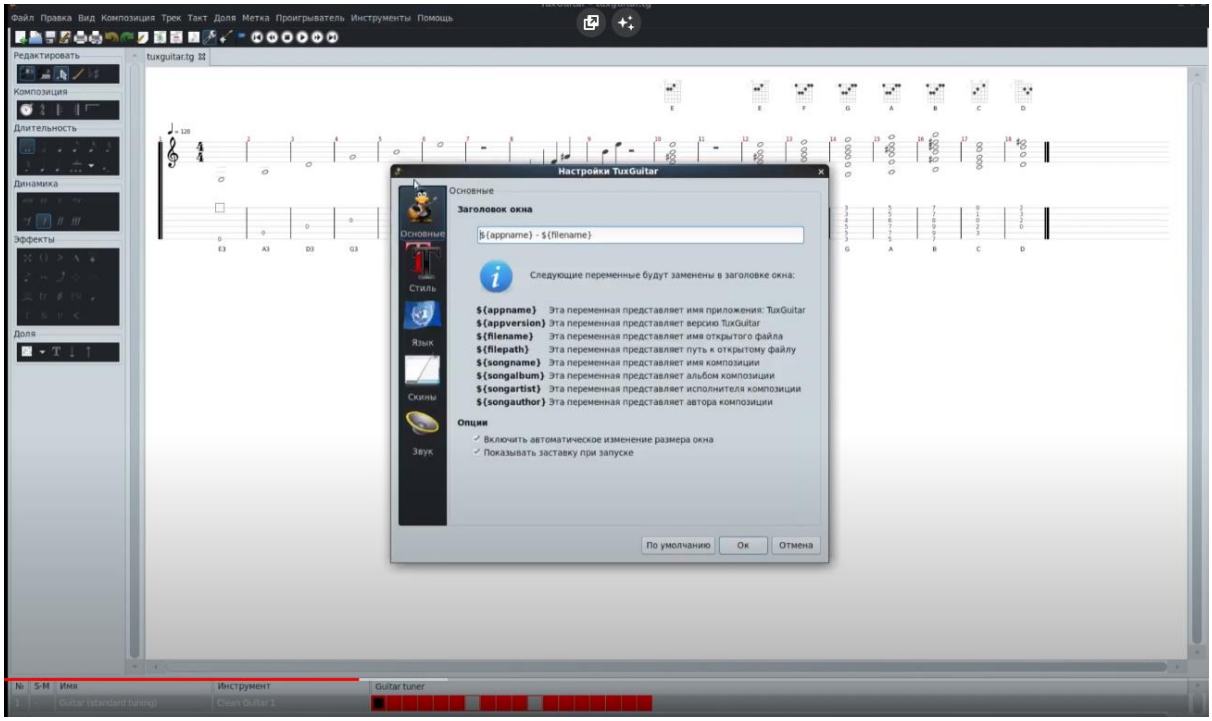

Рисунок 1.5– Меню налаштуваннь “ TexGuitar”

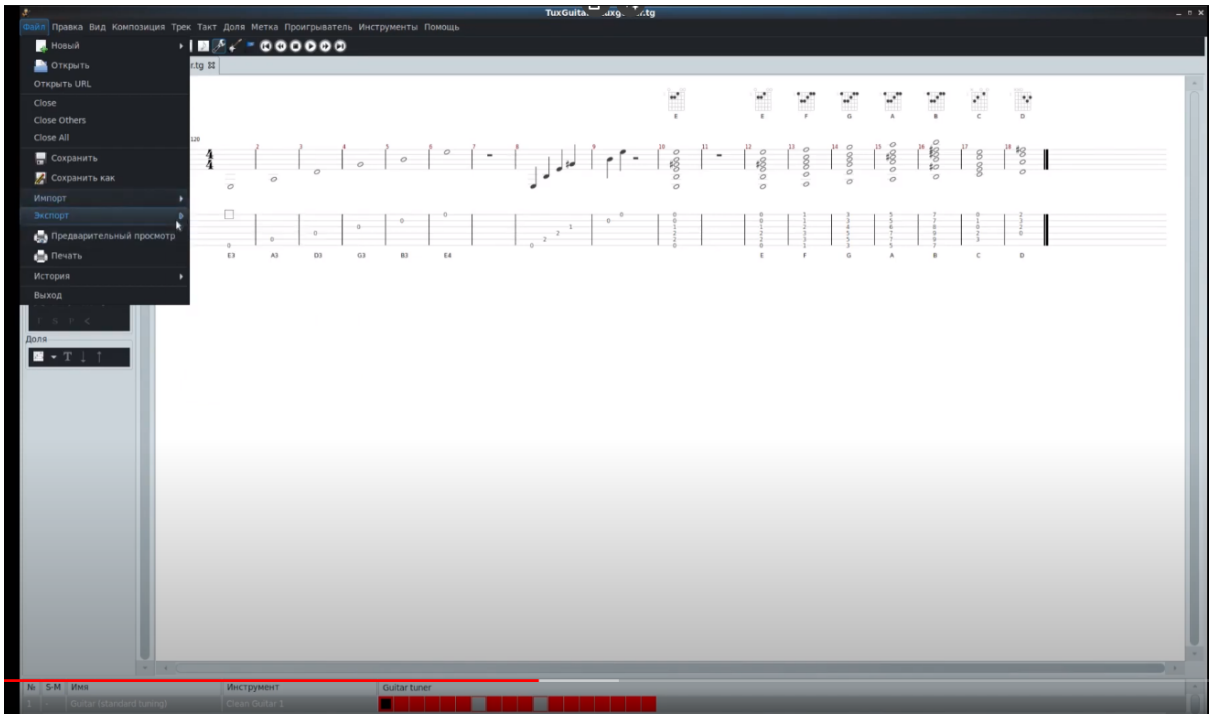

Рисунок 1.6 – Основне меню “ TexGuitar”

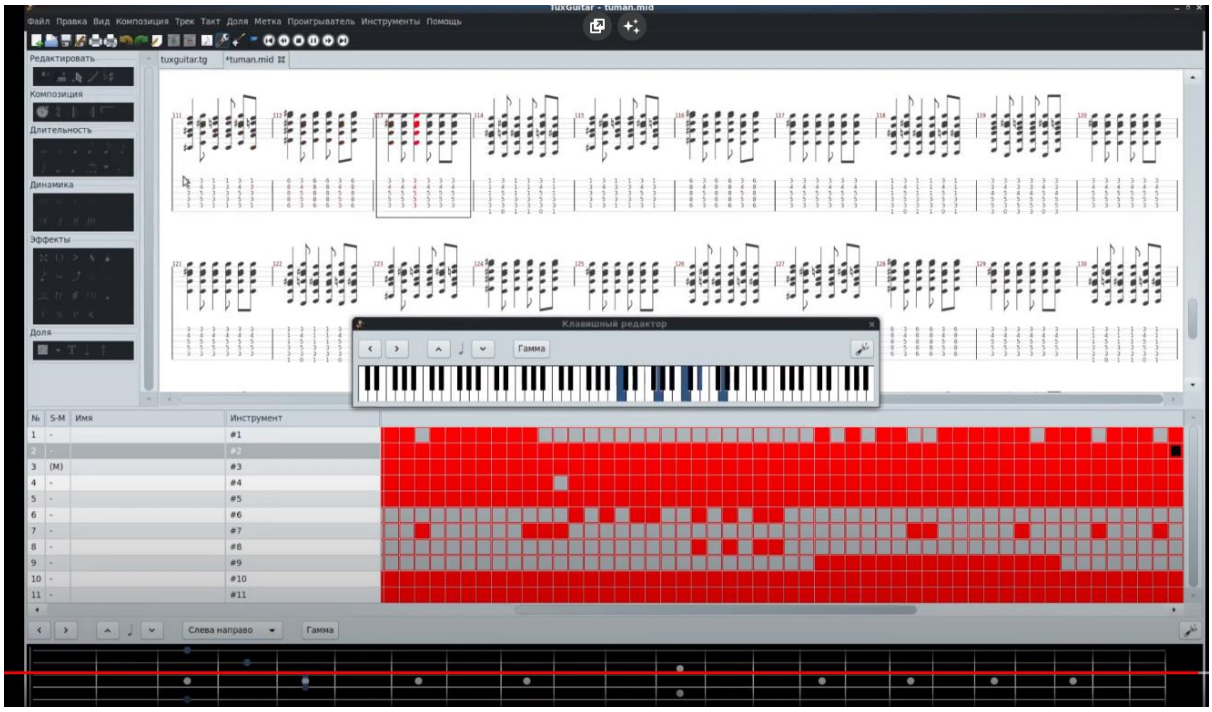

Рисунок 1.7– Можливість обрати інший інструмент “ TexGuitar ”

Після детального аналізу аналогів таблуатурних редакторів, було визначено їх переваги та недоліки. Загалом, можна сказати, що “GuitarPro” випереджає своїх конкурентів у деяких позиціях. Результати аналізу представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів веб-сайтів

Додаток / Характеристика	Guitar Pro	TexGuitar	Songsterr
Сучасний дизайн	+	-	+
Зручний інтерфейс	+	-	+
Інтерактивність	+	+	+
Функціональність	+	-	+
Навігація	+	-	+
Реєстрація користувачів	+	-	+
Можливість додавання та зберігання файлів	+	-	+
Підтримка різних інструментів	+	+	-

1.3 Мета ті задачі дослідження

Мета даного дослідження полягає в розробці програмного додатку для створення та редагування гітарних табулатур, спрямованого на задоволення потреб музикантів та аматорів гри на гітарі. Основною метою є створення зручного та ефективного інструменту для музикантів будь-якого рівня, що дозволить їм створювати, редагувати та відтворювати музику у форматі табулатур.

Основні вимоги до розроблюваного програмного продукту включають:

- Створення інтуїтивного та зручного інтерфейсу для редагування табулатур та нот.

- Реалізація можливості відтворення звуку для перевірки композицій у реальному часі.
- Підтримка широкого спектру інструментів та їх налаштувань для створення різноманітної музики.
- Забезпечення режиму вивчення для навчання гри на гітарі через інтерактивні вправи та режими тренування.

Для досягнення поставлених завдань передбачено такі етапи реалізації:

1. Аналіз сучасного стану ринку програм для створення музики та вивчення їх функціоналу.
2. Проектування інтерфейсу програми, розробка макетів та прототипів для тестування.
3. Реалізація функціоналу програми з використанням мови програмування Java та технологій, що забезпечують зручність і швидкодію додатку.
4. Тестування програмного продукту з метою виявлення та виправлення помилок та недоліків.
5. Оптимізація та доробка програми з урахуванням отриманих результатів тестування.

Для реалізації даного програмного додатку використовуватимуться такі технології та інструменти:

- Мова програмування Java для розробки бекенду програми та логіки її роботи.
- Інструменти для роботи з гітарними табулатурами та аудіофайлами, такі як Apache Commons IO для роботи з файловою системою.
- Бібліотеки для роботи з аудіо, наприклад, Java Sound API для відтворення звуку.

2. МОДЕЛЮВАННЯ ПРОГРАМНОГО ПРОДУКТУ

Розробка включає опис моделей та архітектури системи з використанням відповідних діаграм IDEF0 та UML.

2.1 Моделювання роботи програмного додатку в нотації IDEF0

Моделювання роботи програмного додатку в нотації IDEF0 є важливим етапом, що дозволяє візуалізувати основні функціональні блоки системи та їх взаємозв'язки. Нотація IDEF0 широко використовується для опису процесів та систем, забезпечуючи структурований підхід до аналізу та проектування складних систем.

Процес моделювання розпочався з визначення контекстної діаграми A-0, яка відображає загальний контекст системи та її основні функції. Основний процес системи "Управління табулатурами" включає такі підпроцеси: створення табулатур, редагування табулатур, збереження табулатур, відтворення табулатур та вибір інструменту. Ця діаграма допомагає зрозуміти загальну структуру системи та її взаємодію з зовнішніми системами та користувачами.

Контекстна діаграма IDEF0 програмного додатку зображена на рисунку 2.1

Рисунок 2.1 – Контекстна діаграма основного процесу програмного додатку

На наступному етапі було створено декомпозиційну діаграму A-1, яка деталізує основні функції системи на більш дрібні підфункції. Декомпозиція дозволяє більш детально розглянути функціональність кожного процесу та зрозуміти взаємозв'язки між ними. Наприклад, процес "Створення табулатур" включає введення нових нот та акордів, їх розміщення на відповідних струнах та ладах. Процес "Редагування табулатур" передбачає зміну існуючих нот, включаючи їх переміщення, видалення або заміну. Процес "Збереження табулатур" полягає в збереженні створених або відредагованих табулатур у файл для подальшого використання. Процес "Відтворення табулатур" включає відтворення нот за допомогою MIDI синтезатора, що дозволяє перевірити правильність та якість музичного твору. Процес "Вибір інструменту" дає змогу змінити інструмент для відтворення нот, включаючи акустичну гітару, електрогітару та піаніно.

Діаграми IDEF0 дозволяють візуалізувати вхідні та вихідні дані, механізми та управління, які використовуються в системі. Вхідними даними є "Дані користувача" та "Ноти та акорди", які вводяться користувачем для створення та редагування табулатур. Вихідними даними є "Збережені табулатури" та

"Відтворені табулатури", які зберігаються у файл та відтворюються за допомогою MIDI синтезатора відповідно. Механізми включають JavaFX для створення графічного інтерфейсу користувача, Java Sound API для відтворення нот та файлову систему для збереження та завантаження табулатур. Управління здійснюється через інтерфейс користувача, який забезпечує взаємодію з системою.

Діаграма декомпозиції першого рівня зображена на рисунку 2.2.

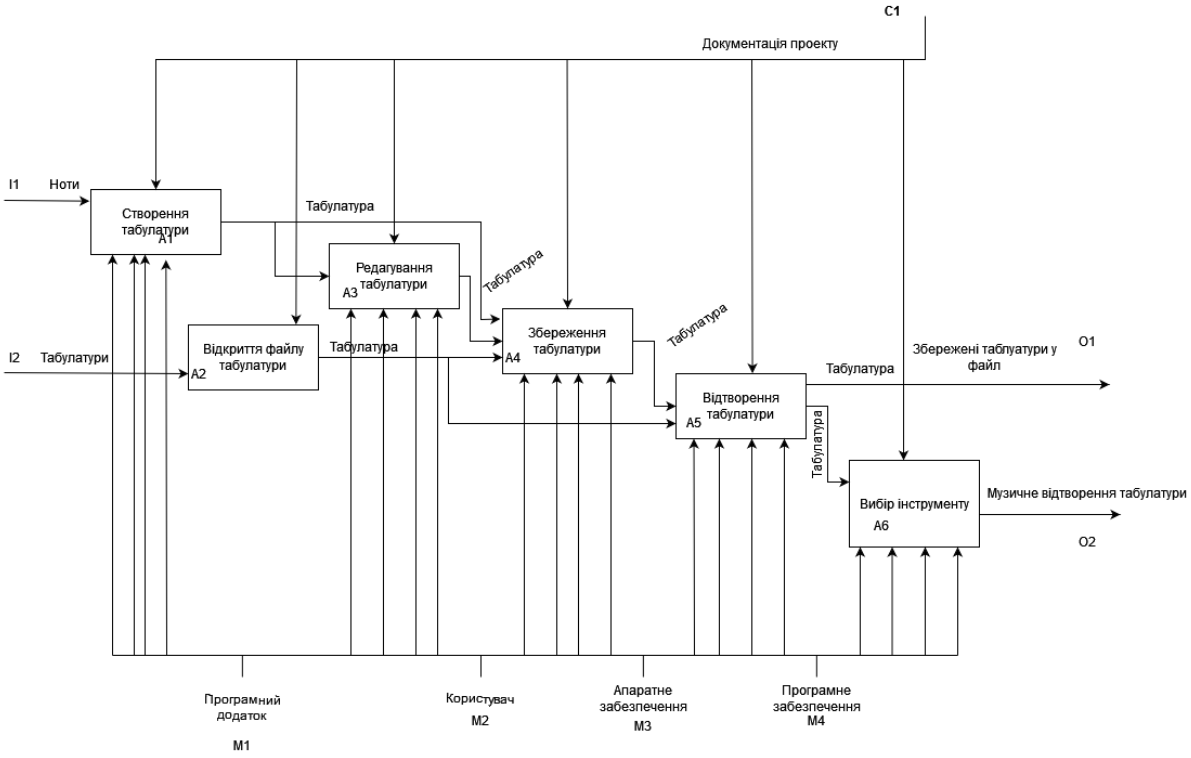

Рисунок 2.2 – Декомпозиція першого рівня основного процесу програмного додатку

2.2 Діаграма варіантів використання програмного додатку

Діаграми варіантів використання (Use Case) є ключовим інструментом для аналізу та проектування взаємодій між користувачами та системою. Вони

дозволяють визначити, як користувачі взаємодіють з системою, які функції вони використовують та які дії виконують. Це допомагає розробникам зрозуміти вимоги до системи та забезпечити її відповідність потребам користувачів.

Для нашого програмного додатку, який створює та редагує гітарні табулатури, було розроблено кілька основних варіантів використання. Перший варіант використання - "Створення нової табулатури". Користувач натискає кнопку "Нова табулатура", після чого відкривається нове вікно для введення нот. Користувач вибирає струну та лад, вводить відповідні ноти або акорди, розміщуючи їх на відповідних позиціях.

Другий варіант використання - "Редагування існуючої табулатури". Користувач відкриває збережену табулатуру для внесення змін, таких як додавання нових нот, переміщення або видалення існуючих нот. Цей процес дозволяє користувачу швидко та зручно вносити необхідні зміни до своїх музичних творів.

Третій варіант використання - "Збереження табулатури". Після створення або редагування табулатури користувач натискає кнопку "Зберегти", вибирає місце для збереження файлу та його назву. Цей процес забезпечує збереження створених або відредагованих табулатур у файл для подальшого використання або обміну з іншими користувачами.

Четвертий варіант використання - "Відтворення табулатури". Користувач натискає кнопку "Програти", і програма відтворює введені ноти за допомогою обраного інструменту. Це дозволяє користувачу почути, як звучить їх музичний твір, та перевірити правильність нотного запису.

П'ятий варіант використання - "Вибір інструменту". Користувач вибирає інструмент зі списку, що впливає на звук при відтворенні нот. Це дозволяє почути різні варіанти звучання музичного твору та обрати найбільш підходящий інструмент для відтворення.

Діаграми варіантів використання допомагають візуалізувати та краще зрозуміти функціональність та взаємодії користувачів з програмним додатком, що є ключовим для його успішної розробки та впровадження. Вони дозволяють

забезпечити відповідність системи вимогам користувачів та зробити її більш зручною та інтуїтивною у використанні.

Діаграма варіантів використання програмного додатку зображена на рисунку 2.3.

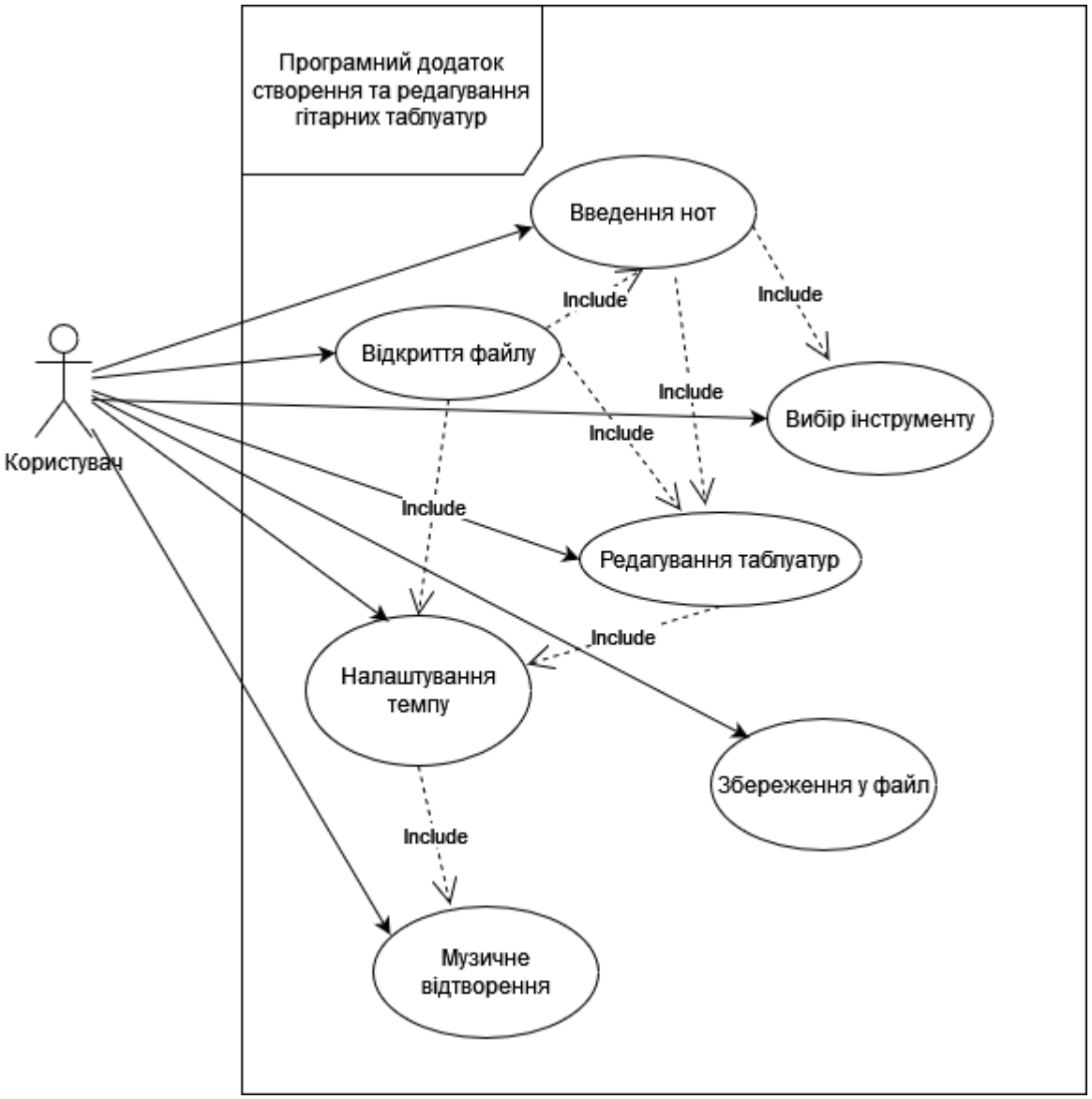

Рисунок 2.3 – Діаграма варіантів використання програмного додатку

На основі розроблених діаграм IDEF0 та діаграм варіантів використання було створено програмний додаток, який відповідає вимогам початкових музикантів та дозволяє їм легко та ефективно працювати з гітарними табулатурами. Додаток поєднує в собі простоту використання та функціональність, що робить його

корисним інструментом для музикантів-початківців. Завдяки використанню сучасних методів моделювання вдалося досягти високої якості програмного продукту, що відповідає вимогам користувачів і дозволяє їм ефективно працювати з музичними композиціями.

3. РОЗРОБКА ПРОГРАМНОГО ПРОДУКТУ

Редактор табулатур розроблено на основі мови програмування Java з використанням бібліотеки JavaFX для створення графічного інтерфейсу користувача. Для забезпечення можливості відтворення нот було застосовано Java Sound API.

Основні компоненти системи включають:

- Графічний інтерфейс користувача (GUI): Використання JavaFX дозволило створити інтуїтивно зрозумілий інтерфейс, який забезпечує зручний доступ до функцій програми. GUI складається з таких елементів, як текстові поля для введення нот, кнопки для виконання основних операцій (збереження, відкриття, програвання табулатур), а також інструменти для налаштування параметрів (вибір інструменту, BPM).
- Модель зберігання даних: Нотні записи зберігаються у вигляді текстових файлів, що полегшує процес імпорту та експорту табулатур. Кожен рядок файлу відповідає окремій струні гітари, а кожен стовпець - окремій ноті.
- Модель програвання звуку: Бібліотека Java Sound API використовується для відтворення нот за допомогою MIDI синтезатора. Це забезпечує високу якість звуку та підтримку різних інструментів.

3.1.1 Програмна реалізація табулатурного редактору

Після запуску IDE створюється новий проект JavaFX, для якого обирається назва проекту, розміщення та підключається JDK (комплект розробника, який включає в себе компілятор Java, стандартні бібліотеки класів, документацію, тощо).

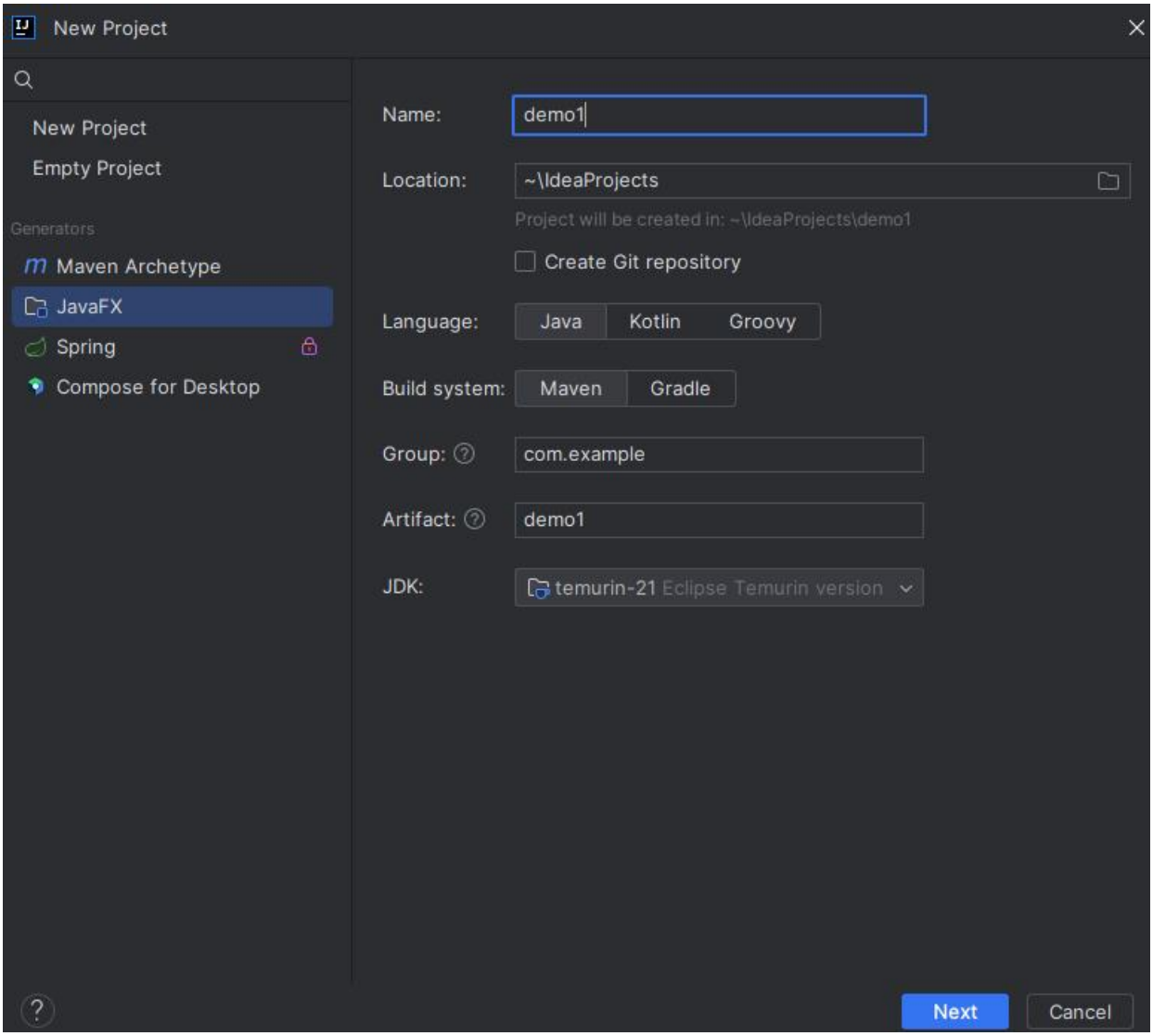

Рисунок 3.1 – Створення проекту

У результаті проведеної роботи було створено шаблон проекту з правильною файловою структурою, спеціально орієнтованою для цього програмного додатку. Створений шаблон забезпечує зручну організацію та управління файлами, що полегшує подальшу розробку та підтримку програми.

Файлова структура проекту включає основні компоненти, необхідні для функціонування додатку. Це забезпечує чіткий розподіл відповідальності між різними частинами коду, сприяючи кращій модульності та повторному використанню коду

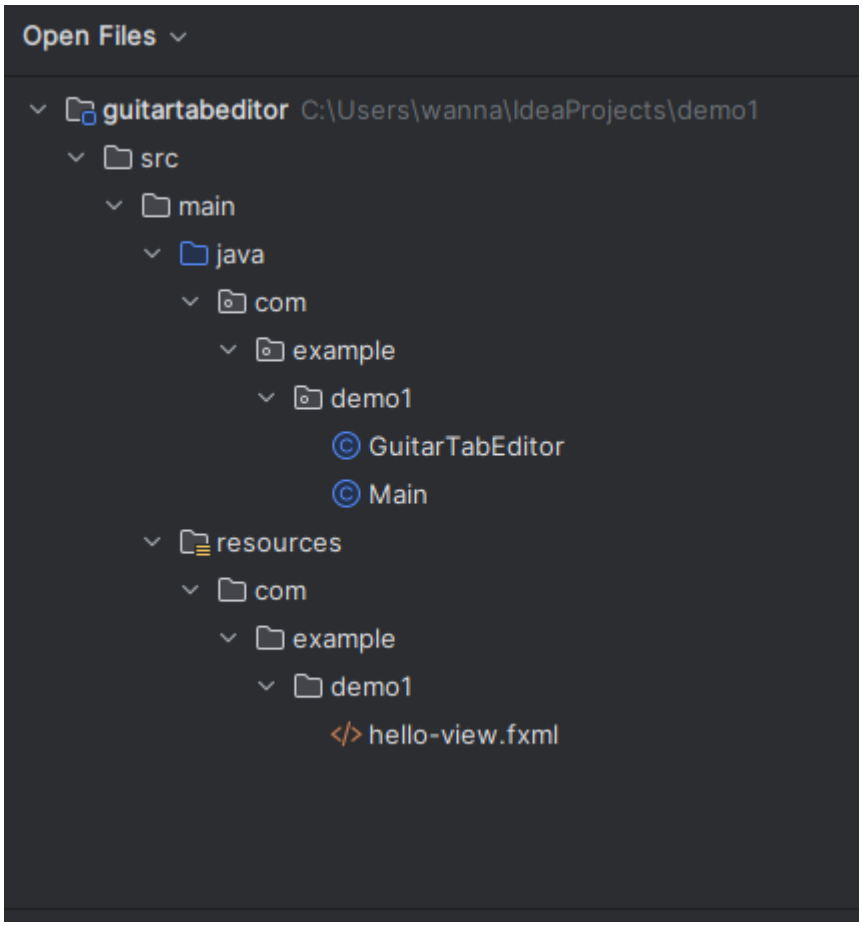

Рисунок 3.2 – Структура проекту

У створеній структурі проекту присутні три файли, готові для заповнення: Main.java, GuitarTabEditor.java, та hello-view.fxml. Основна частина програмного коду описана у файлі контролера GuitarTabEditor.java, файл типу .fxml відповідає за інтерфейс, а Main.java відповідає за запуск програми та ініціалізацію головного вікна програмного додатку.

У процесі розробки програмного додатку було заповнено файли проекту програмними кодами, написаними на мові програмування Java. Кожен з цих файлів виконує специфічну функцію, забезпечуючи зручність використання та функціональність додатку.

Файл Main.java містить основний клас, який відповідає за запуск програми. У ньому описано метод main, що ініціалізує головне вікно додатку та завантажує

інтерфейс з файлу hello-view.fxml. Цей файл також містить налаштування для початкової конфігурації програми та визначення параметрів запуску.

Файл GuitarTabEditor.java виконує роль контролера. Він містить методи для обробки взаємодій користувача з інтерфейсом, керує логікою програми та забезпечує зв'язок між інтерфейсом користувача і моделлю. Цей файл включає методи для створення, редагування, збереження та відтворення табулатур. Зокрема, методи createTab, editTab, saveTab та playTab реалізують відповідні функції додатку, забезпечуючи зручність роботи для користувача.

Файл hello-view.fxml містить опис інтерфейсу користувача на мові розмітки FXML. У ньому визначено компоненти інтерфейсу, такі як кнопки, текстові поля, таблиці та інші елементи, які відображаються на екрані користувача. Файл FXML дозволяє відокремити логіку програми від її візуального представлення, що сприяє кращій організації коду та спрощує його підтримку.

У результаті розробки програмного додатку було заповнено файли проекту програмними кодами, що забезпечують повну функціональність системи. Опис заповнених програмним кодом файлів наведено у таблиці 3.1

Таблиця 3.1 – Опис файлів програмного додатку

Файл	**Опис файлу**
Main.java	відповідає за запуск JavaFX застосунку та ініціалізацію головного вікна програми
GuitarTabEditor.java	клас контролера, який керує логікою роботи інтерфейсу
hello-view.fxml	визначає структуру інтерфейсу користувача

Для реалізації функціоналу створимо такі методи у основному файлі GuitarTabEditor.java які забезпечать функціональність редактора створення та редагування гітарних табулатур. Створені методи описано у таблиці 3.2

Таблиця 3.2 Методи файлу GuitarTabEditor.java та їх опис

Назва методу	**Опис методу**
initialize	Метод відповідає за початкове налаштування інтерфейсу користувача. Під час розробки цього методу було враховано необхідність створення табличної структури для введення нот. Метод було розроблено з використанням JavaFX для забезпечення автоматичного налаштування інтерфейсу під час запуск програми. Також метод встановлює обробники подій для елементів інтерфейсу, таких як текстові поля та кнопки.
newTab	Метод реалізую створення нової табулатури шляхом очищення всіх текстових полів в таблиці готуючи для введення нових нот. Основною метою цього методу було забезпечення можливості швидко почати нову композицію без необхідності перезавантаження програми
openTab	Цей метод забезпечує завантаження табулатури з текстового файлу. Цей метод використовує стандартний діалог вибору файлів для вибору файла користувачем, після чого зчитує вміст файлу та заповнює відповідні текстові поля на інтерфейсі. Розробка цього методу включала обробку помилок при зчитуванні файлів та перевірку коректності даних що завантажуються.
saveTab	Метод забезпечує збереження введених нот у текстовий файл. Він відкриває діалог збереження файлу, де користувач може вибрати місце для збереження, після чого записує вміст таблиці у файл. Важливим аспектом розробки цього методу було забезпечення коректного формату збереження даних, щоб їх можна було легко завантажити в майбутньому.

Продовження таблиці 3.2

playTab	Відповідає за відтворення нот, введених у табличну форму. Для реалізації цього методу було використано Java Sound API, що дозволяє відтворювати MIDI сигнали. Основною складністю при розробці цього методу була синхронізація відтворення нот відповідно до значення BPM (ударів на хвилину), заданого користувачем.
AddColumn	Метод забезпечує динамічне додавання нового стовпця до таблиці нот. Це дозволяє користувачам розширювати табулатуру за необхідності. Під час розробки цього методу враховувалася необхідність збереження коректного відображення та функціональності таблиці після додавання нових елементів.
UpdateInstrument	Метод відповідає за зміну поточного музичного інструменту, який використовується для відтворення нот, відповідно до вибору користувача в ComboBox. Це дозволяє користувачам змінювати звук табулатур між акустичною гітарою, піаніно та електрогітарою. Для реалізації цього методу використовувалися можливості JavaFX, зокрема, обробка подій вибору у ComboBox. При виборі нового інструменту метод `updateInstrument` змінює поточний інструмент, який використовується для відтворення нот, на основі вибраного значення у ComboBox.

Всі методи були розроблені з урахуванням вимог до функціональності програми та з використанням можливостей JavaFX та Java Sound API. Важливим аспектом розробки було забезпечення зручності використання інтерфейсу та

коректної обробки введених даних, що дозволило створити ефективний та надійний інструмент для роботи з гітарними табулатурами.

Файл Main.java є головним файлом програми, який відповідає за її запуск та ініціалізацію головного вікна з використанням JavaFX. Основна мета цього файлу полягає у створенні початкового налаштування інтерфейсу користувача та відображенні вікна програми на екрані. Метод start було розроблено з урахуванням вимог до зручності використання та надійності роботи програми. Він забезпечує коректне завантаження та відображення інтерфейсу користувача, що є важливим для забезпечення позитивного досвіду користувача.

Відповідні компоненти та їх опис наведені у таблиці 3.3

Таблиця 3.3 Компоненти файлу Main.java та їх опис

Компонент	**Опис**
Завантаження FXML файлу	Метод використовує FXMLLoader для завантаження файлу hello-view.fxml, який визначає структуру інтерфейсу користувача. Це дозволяє розділити логіку програми та її інтерфейс, що полегшує розробку та підтримку коду.
Створення сцени	Після завантаження FXML файлу створюється об'єкт Scene, який представляє вміст вікна програми. Сцена встановлюється на головній стадії (Stage), яка відповідає за відображення вікна програми на екрані.
Встановлення заголовка	Метод встановлює заголовок вікна, який відображається на верхній панелі вікна програми.
Відображення вікна	Після виконання всіх налаштувань метод викликає stage.show(), що відображає головне вікно програми на екрані.

Файл hello-view.fxml визначає структуру графічного інтерфейсу користувача з використанням FXML (FXML Markup Language). Цей файл містить опис усіх елементів інтерфейсу, таких як кнопки, текстові поля, таблиці та інші компоненти.

Основні компоненти FXML файлу наведені у таблиці 3.4

Таблиця 3.4 Компоненти файлу hello-view.fxml та їх опис

Компонент	**Опис**
MenuBar	Включає меню "File" з пунктами "New", "Open", "Save" та "Exit". Це забезпечує доступ до основних функцій програми через зручний інтерфейс.
HBox	Горизонтальна панель, яка містить кнопки "New", "Open", "Save", "Play", "Add Column" та інші елементи управління, такі як поле для введення BPM та випадаючий список для вибору інструменту. Ця панель забезпечує зручний доступ до основних функцій програми.
ScrollPane	Обгортка для вертикальної панелі (VBox), яка містить таблицю нот (GridPane). Це дозволяє користувачу прокручувати таблицю, якщо вона перевищує розміри вікна програми.
GridPane	Таблиця для введення нот, яка містить текстові поля для кожної струни гітари. Кожне текстове поле відповідає окремій ноті.

Розробка FXML файлу включала наступні етапи:

- Створення структури інтерфейсу: спочатку було визначено основні компоненти інтерфейсу та їх розташування на екрані. Це включало розміщення MenuBar у верхній частині вікна, HBox з кнопками та іншими елементами управління під MenuBar, та ScrollPane, що містить GridPane для введення нот.

- Налаштування властивостей компонентів: для кожного компонента було налаштовано необхідні властивості, такі як розміри, відступи, стилі та обробники подій. Наприклад, для кнопок було визначено обробники подій onAction, які викликають відповідні методи контролера.
- Тестування: після початкового створення файлу було проведено тестування для перевірки коректності роботи інтерфейсу. Виявлені помилки та неточності було виправлено.

Повний лістинг програмних кодів з коментарями наведений у окремому додатку (Додаток В)

3.2 Використання програмного додатку

При запуску програмного додатку з’являється головне вікно з областю для внесення нот та функціональним меню (Рис. 3.3 – Головне вікно програми).

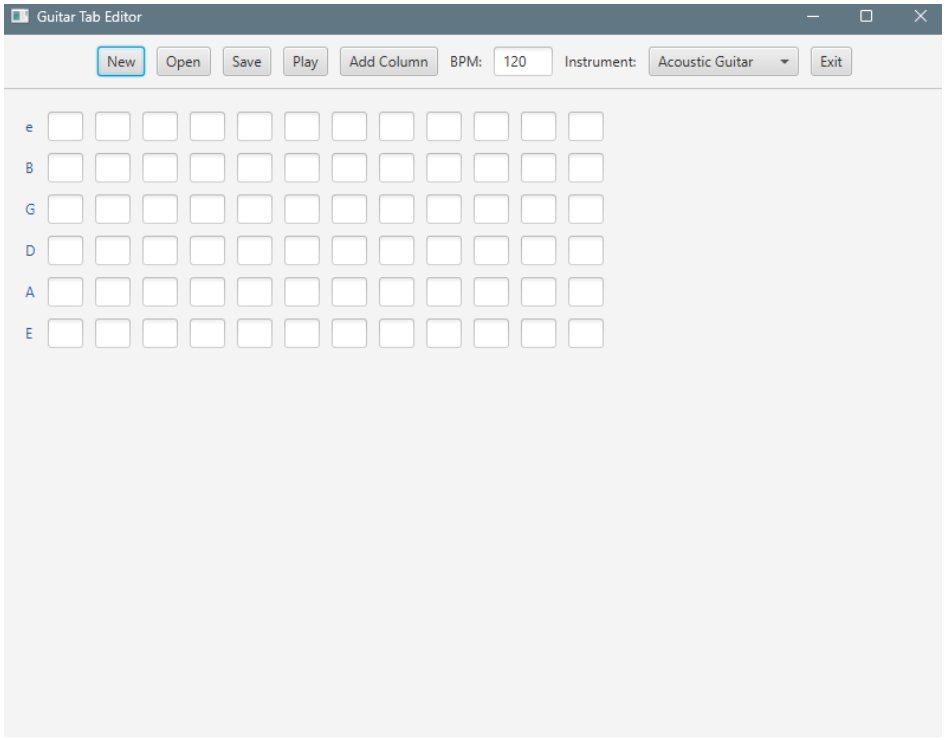

Рисунок 3.3 – Головне вікно програми

У цьому вікні користувач бачить основні елементи інтерфейсу, такі як меню, кнопки для основних дій, робочу область для введення нот та інструменти для редагування табулатур.

Для початку роботи з додатком користувачу слід почати вводити номери гітарного ладу у відповідні поля на робочій області (Рис. 3.4 – Введення нот для створення табулатури).

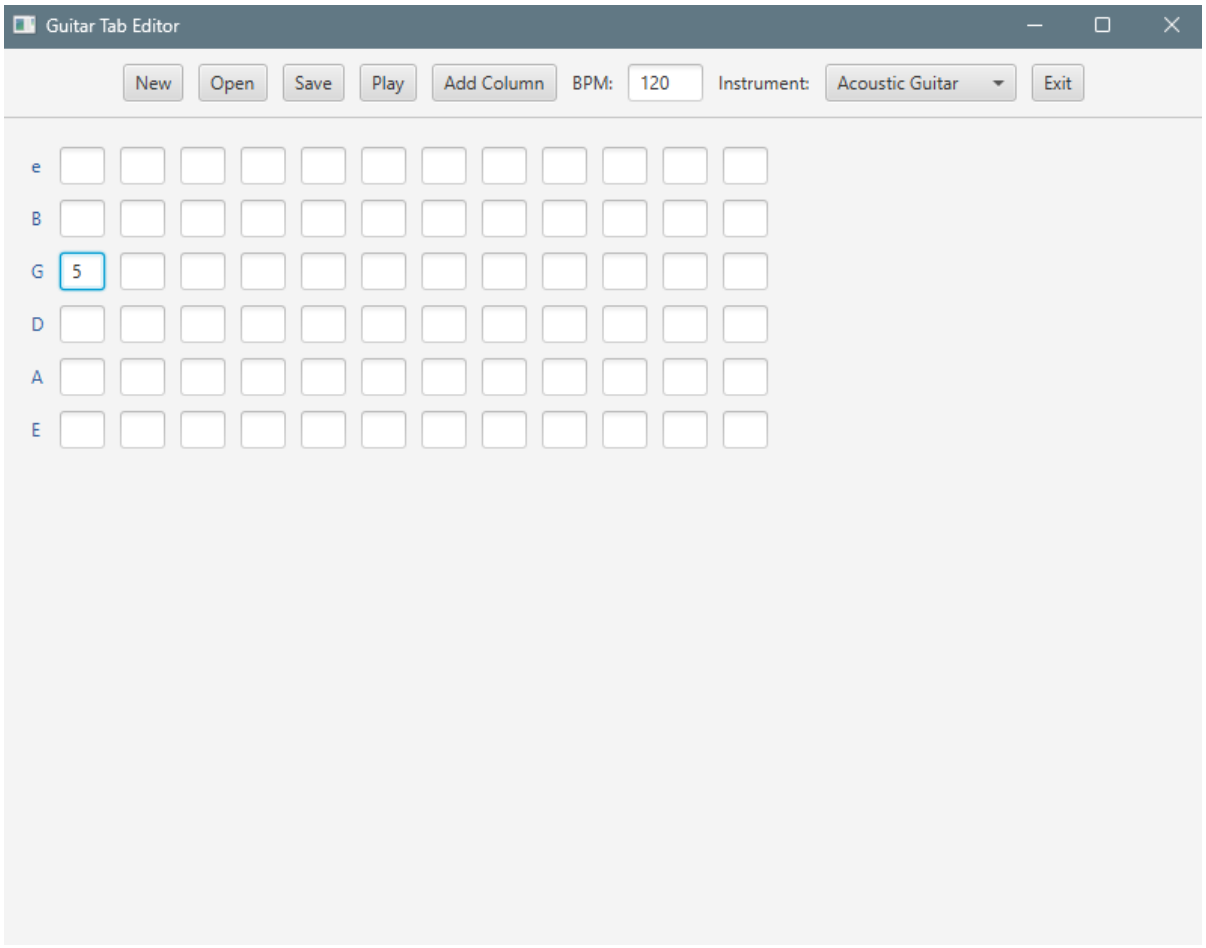

Рисунок 3.4 – Введення нот для створення табулатури

Кожен рядок робочої області відповідає окремій струні гітари. Користувач може вибрати будь-яку струну та ввести номер ладу, на якому він хоче поставити ноту. Наприклад, якщо користувач хоче поставити ноту на третій струні на п'ятому ладу, він вводить "5" у відповідне поле третього рядка.

Також початком роботи є відкриття вже створеного файлу з написаною у ньому табулатурою. Для цього на верхній панелі є відповідна кнопка "Open" (Рис. 3.5 – Відкриття файлу зі створеною табулатурою). Натискання цієї кнопки відкриває діалогове вікно вибору файлу, де користувач може знайти і вибрати необхідний файл з табулатурою. Після відкриття файлу, табулатура відобразиться у робочій області, і користувач зможе продовжити її редагування або відтворення.

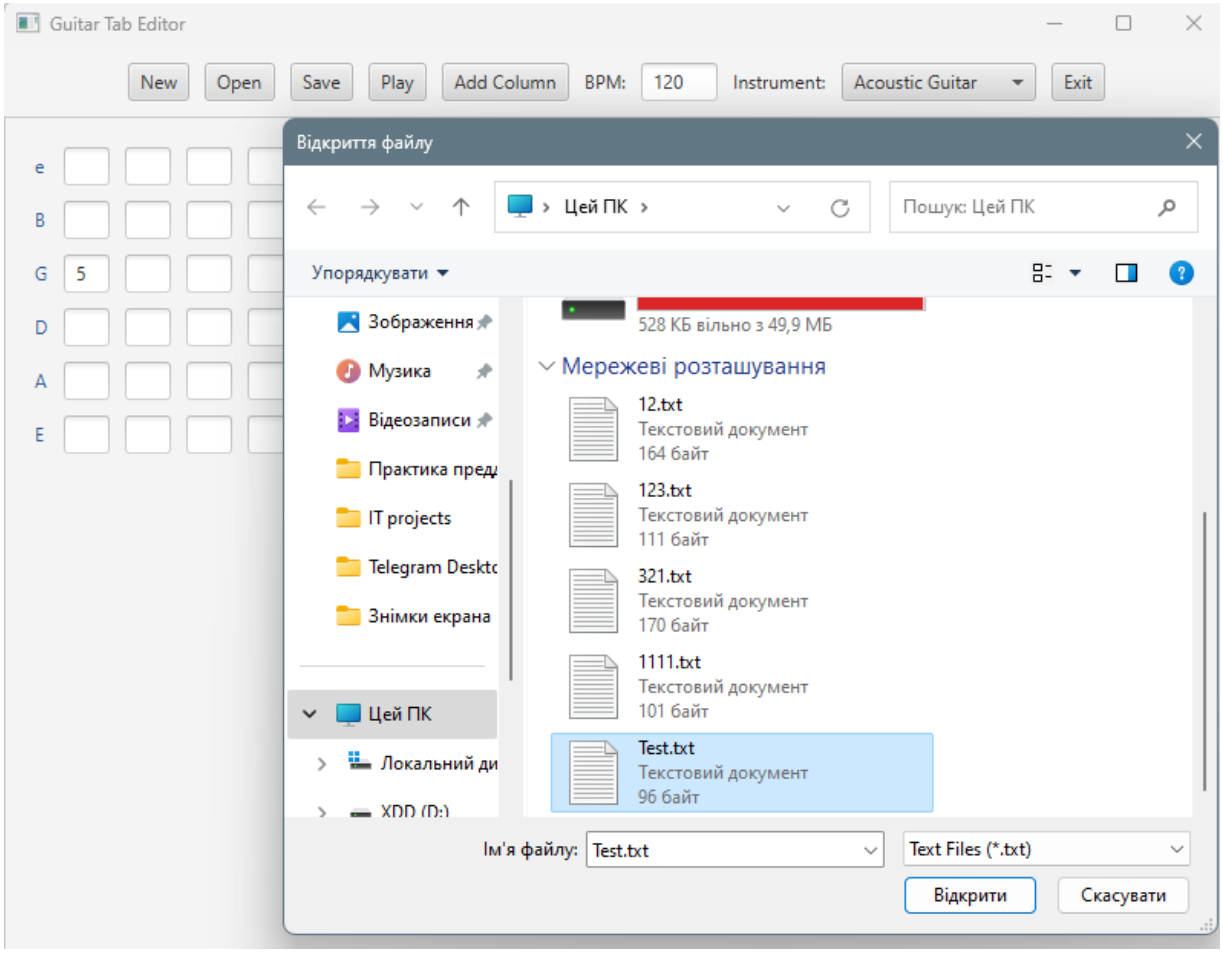

Рисунок 3.5 – Відкриття файлу зі створеною табулатурою

Після введення нот користувач може переглянути і перевірити правильність своєї табулатури. Якщо необхідно внести зміни, користувач може легко редагувати введені ноти, просто змінюючи номери ладів у відповідних полях. Кожна зміна автоматично зберігається у робочій області, що дозволяє користувачеві швидко і ефективно редагувати табулатуру.

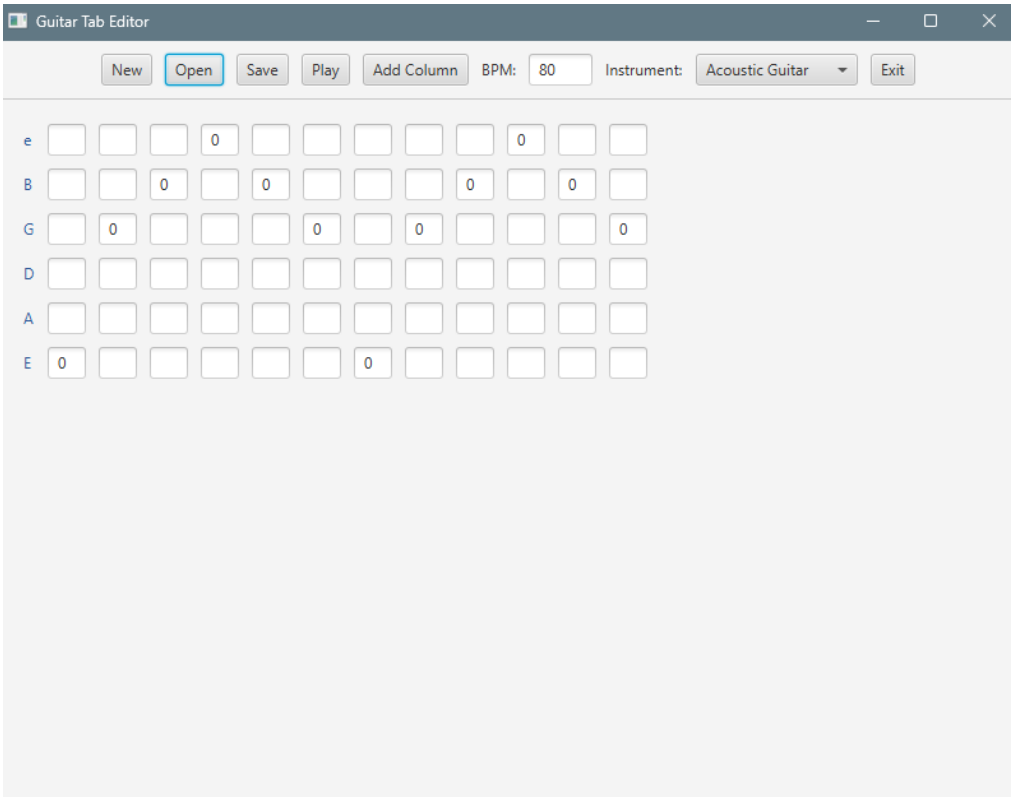

Рисунок 3.6 – Табулатура з відкритого файлу

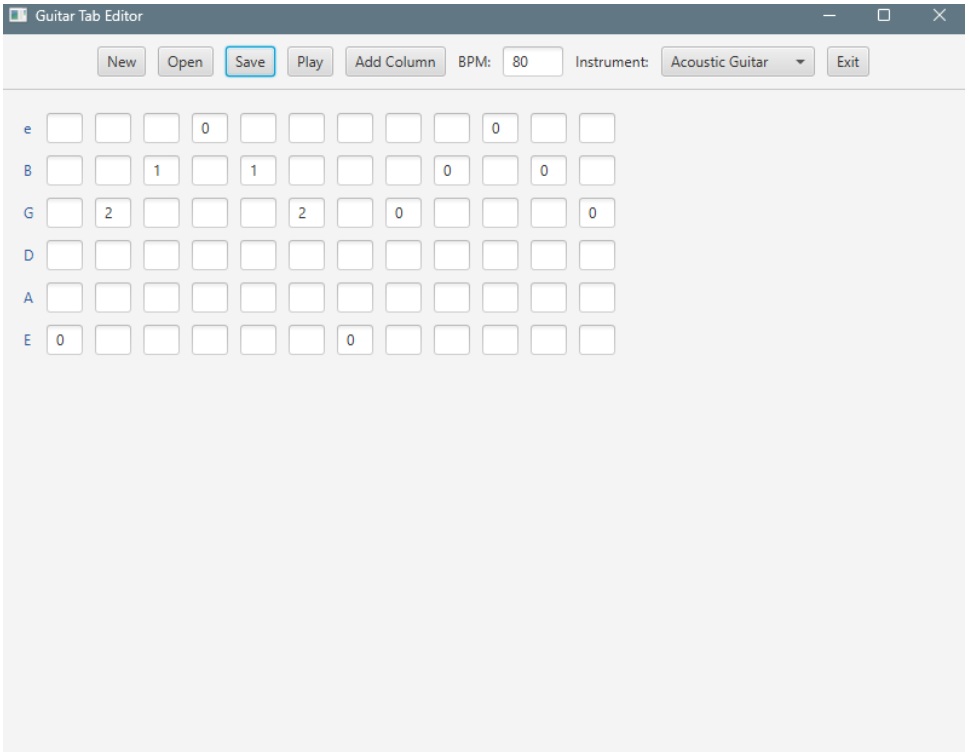

Рисунок 3.7 – Редагування табулатури відкритого файлу

При написанні свого варіанту мелодії користувачу варто зберегти створений матеріал у файл за допомогою кнопки "Save" (Рис. 3.6 – Збереження у файл). Після натискання цієї кнопки відкривається діалогове вікно, де користувач може вказати назву файлу та місце його збереження. Збережений файл можна передати іншому користувачу, що забезпечує обмін своїми досягненнями. Це особливо зручно для музикантів, які працюють у команді або хочуть поділитися своєю творчістю з іншими.

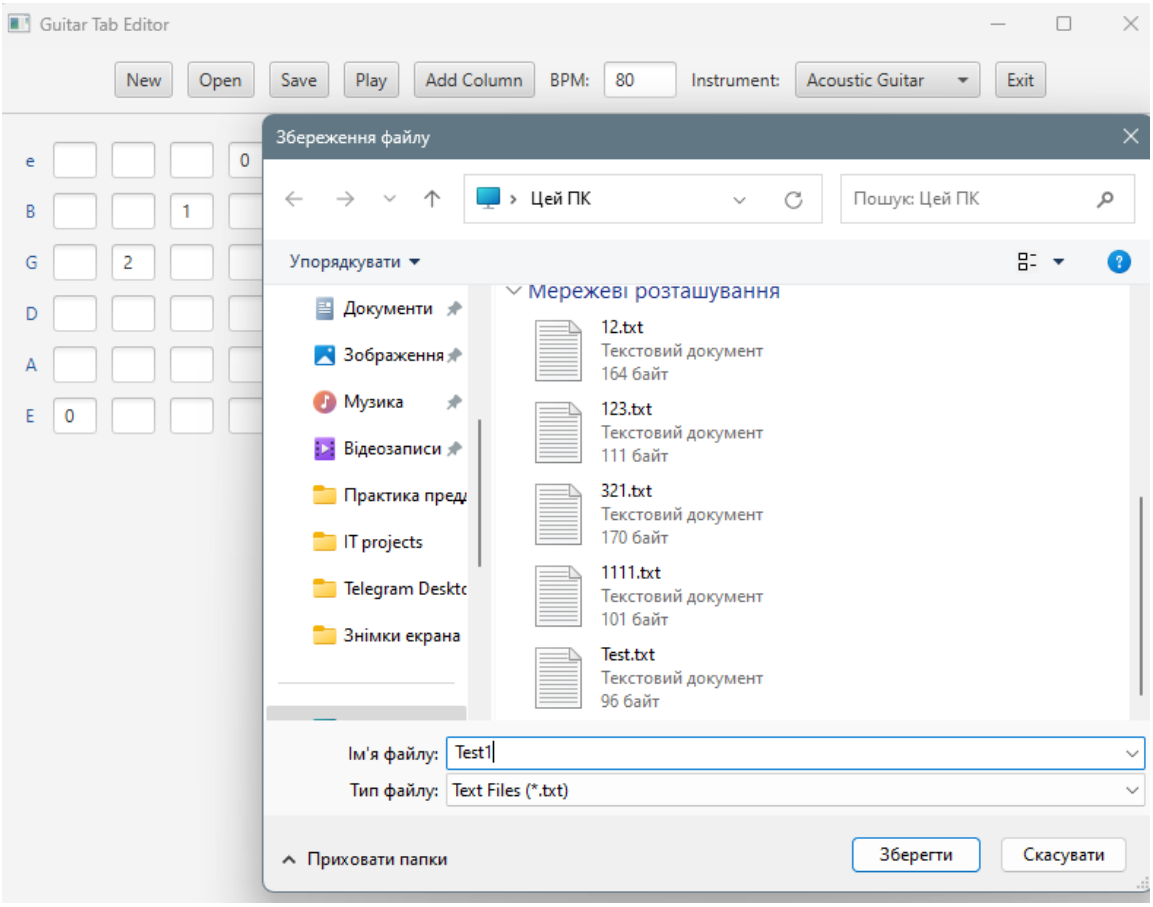

Рисунок 3.8 – Збереження у файл

За допомогою графи "Instrument" користувач може змінити інструмент, на якому відтворюватиметься звук (Рис. 3.7 – Зміна музичного інструменту). Це дозволяє користувачеві експериментувати з різними звуками і знайти найкращий варіант для своєї композиції. Наприклад, користувач може змінити звук з

акустичної гітари на електрогітару або піаніно, що додасть нові можливості для створення музичних творів.

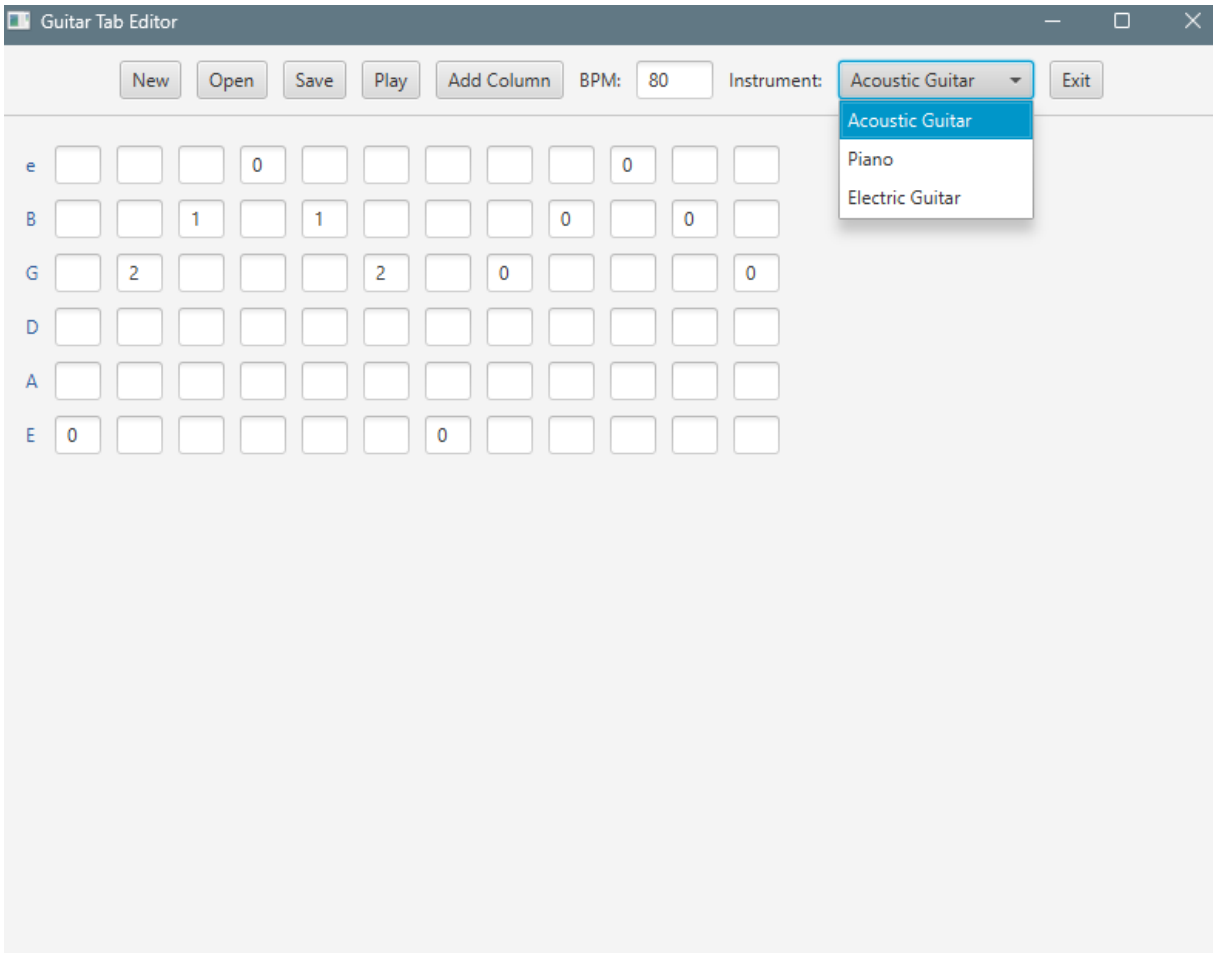

Рисунок 3.9 – Зміна музичного інструменту

Якщо для користувача недостатньо робочої області, він може розширити її за допомогою функції додати рядок (Рис. 3.8 – Додання нового ряду). Натискання цієї кнопки додає новий рядок до робочої області, що дозволяє вводити більше нот і створювати складніші композиції. Це зручно для користувачів, які працюють над довгими або складними музичними творами.

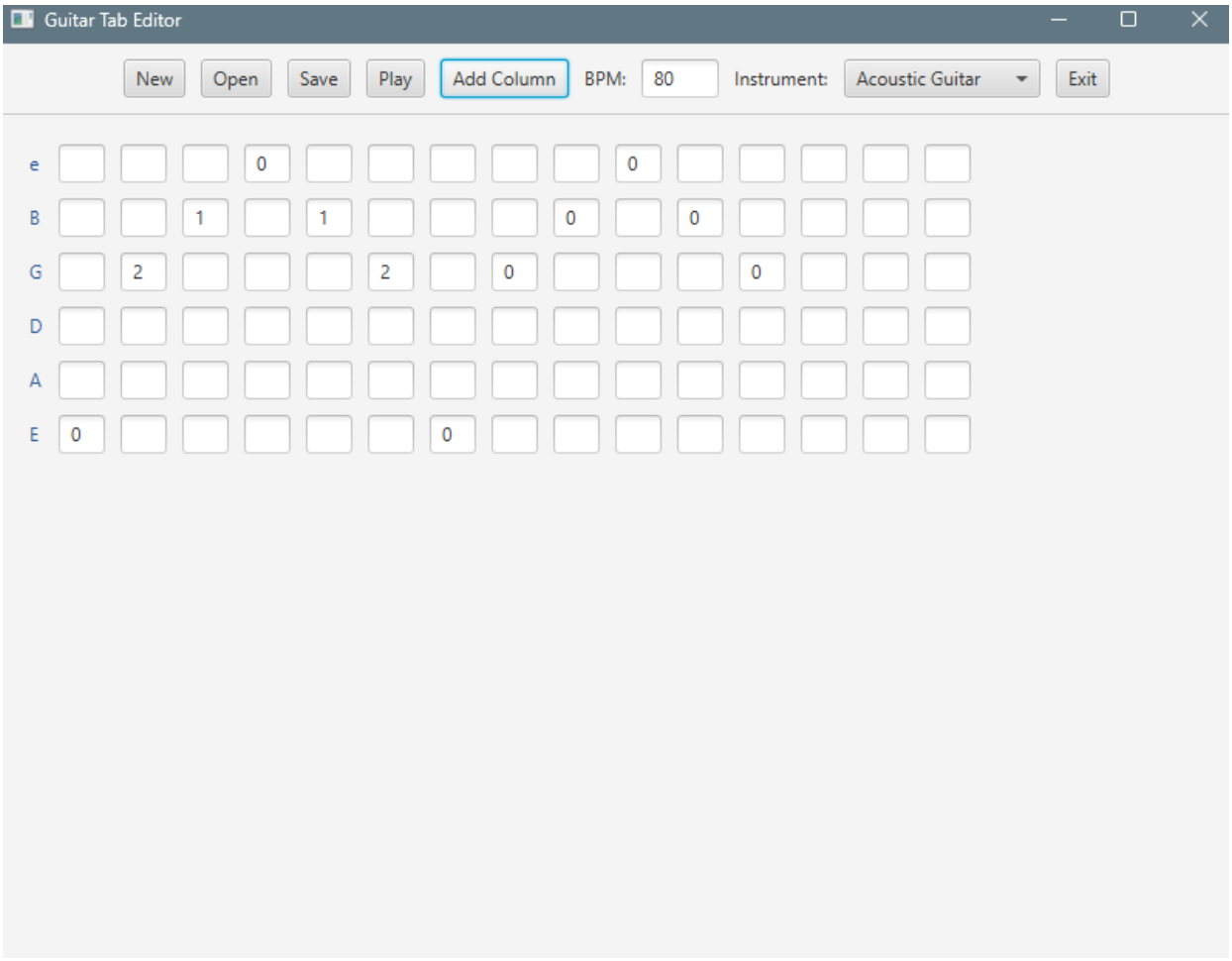

Рисунок 3.10 – Додання нового ряду

Додаток також надає можливість змінювати темп виконання композиції, що є важливим для точного відтворення музичного твору. Для зміни темпу на головній панелі інструментів передбачене спеціальне поле, в якому користувач може ввести значення темпу у бітах за хвилину (BPM) (Рис. 3.9 – Зміна темпу). Це дозволяє налаштувати швидкість відтворення композиції, що особливо корисно при навчанні та практиці гри на гітарі.

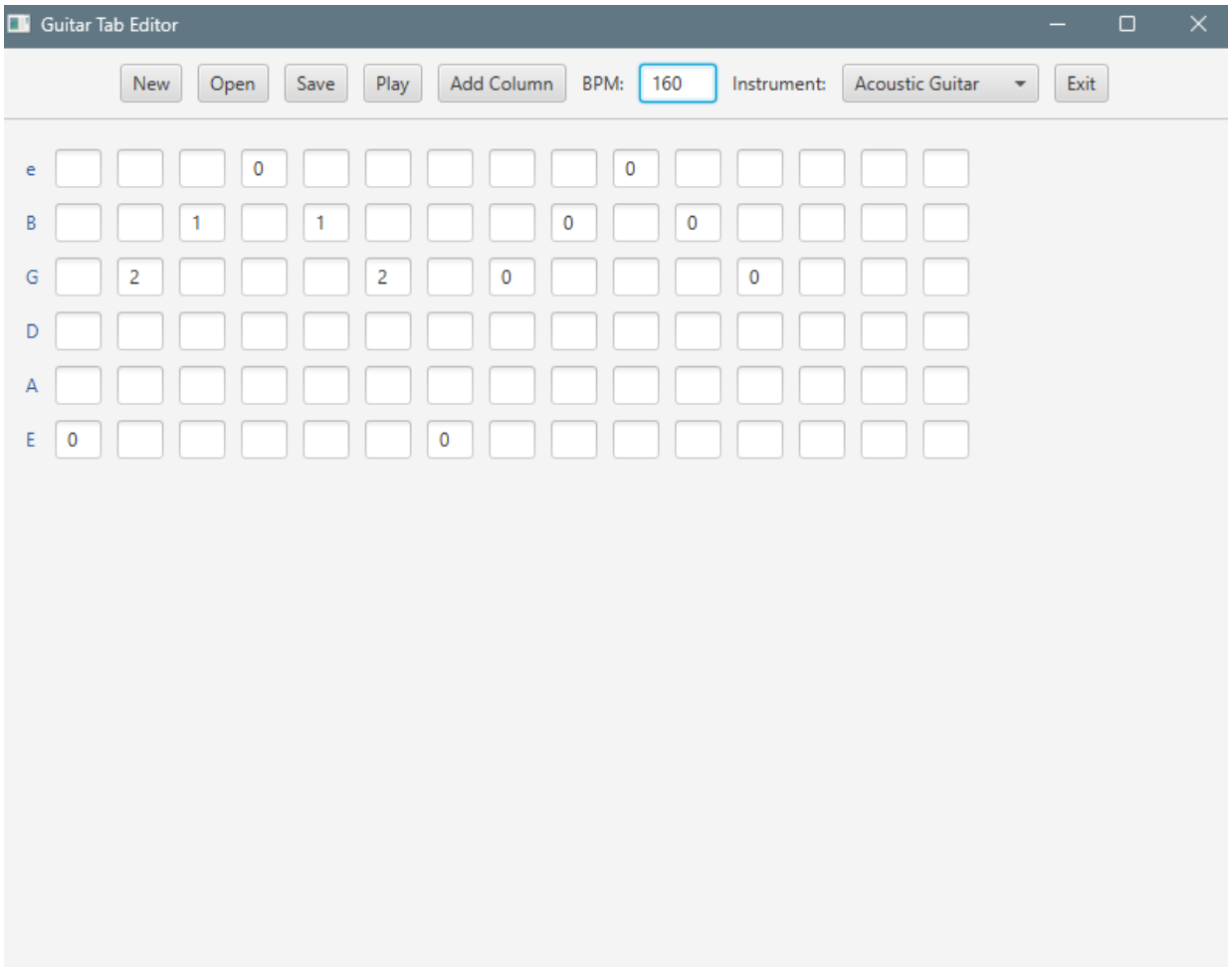

Рисунок 3.11 – Зміна темпу

Користувач може експериментувати з різними значеннями темпу, щоб знайти оптимальний для конкретного твору або вправи. Після введення нового значення темпу програма автоматично налаштовує швидкість відтворення відповідно до зазначеного параметра, що забезпечує точність та комфортність роботи з музичними творами.

Після завершення роботи слід вийти з програми натиснувши кнопку "Exit". Це закриває програму і зберігає всі внесені зміни. Завершення роботи за допомогою цієї кнопки забезпечує правильне закриття всіх відкритих файлів і запобігає можливим втратам даних.

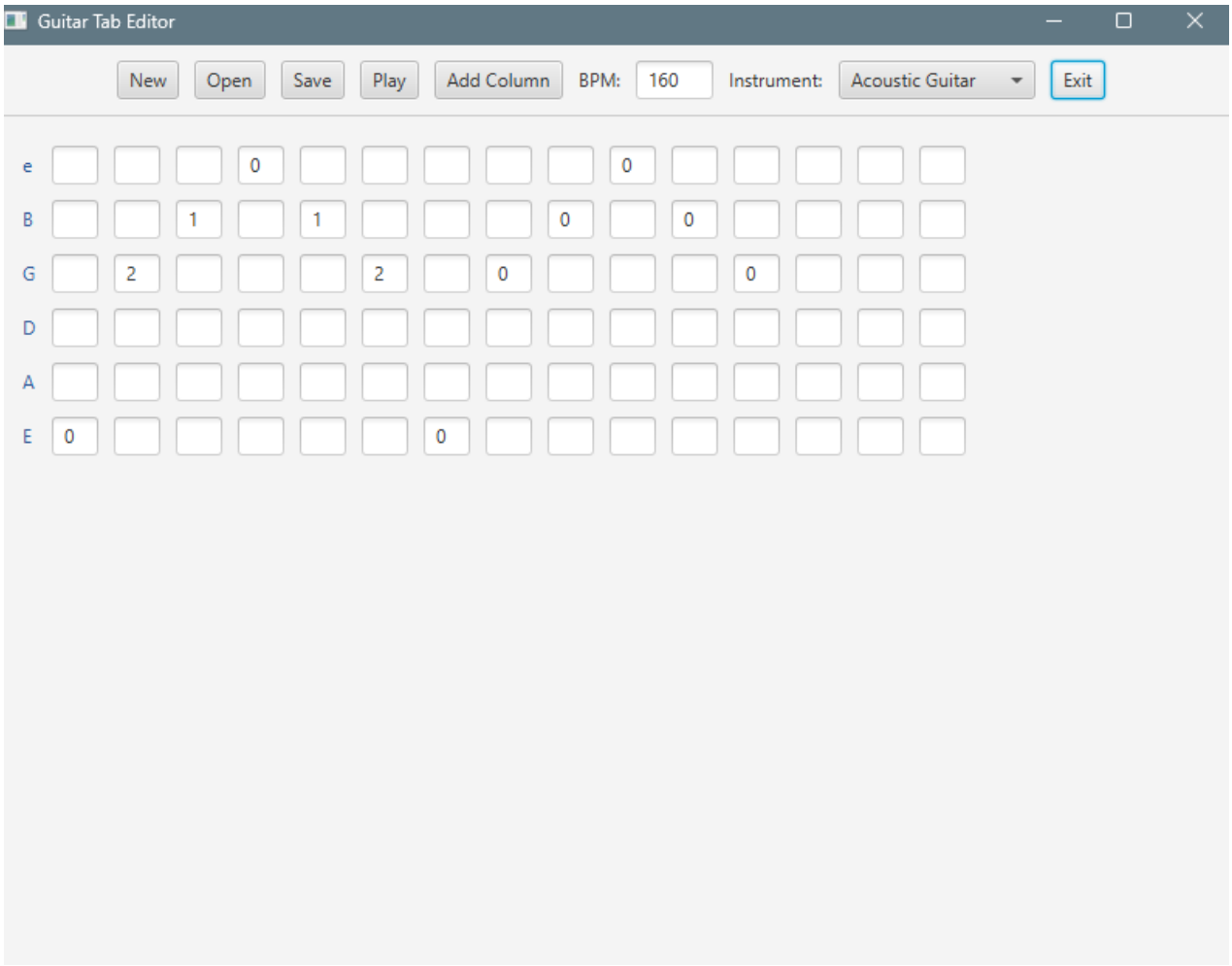

Рисунок 1.12 – Вихід з програми

На рисунку 3.13 – показано вміст файлу табулатури, який демонструє, як зберігаються введені ноти і акорди у файлі. Це важливо для розуміння структури даних та подальшої роботи з файлами табулатур.

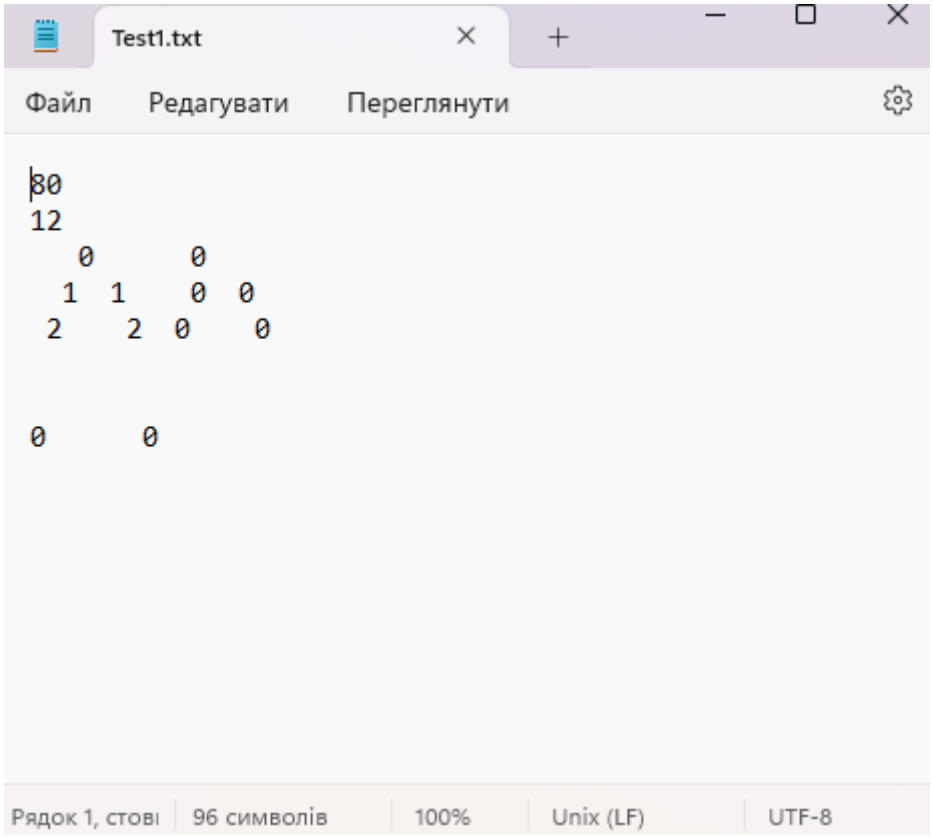

Рисунок 3.13 – Зміст файлу

ВИСНОВКИ

У ході виконання цієї кваліфікаційної роботи бакалавра було проведено детальний аналіз предметної області, визначено ключові проблеми та підтверджено їхню актуальність. Під час дослідження були виявлені два програмні додатки та один веб-додаток, які виконують схожі функції. Результати цього дослідження були систематизовані у вигляді таблиці, що дозволило виявити недоліки та прогалини в існуючих рішеннях, які слід було врахувати при розробці нового додатку. Зроблений аналіз сучасного стану ринку програм для створення музики показав, що існуючі рішення, такі як TuxGuitar та Songsterr, мають широкий функціонал, але водночас можуть бути складними для початківців. Це підтвердило актуальність розробки нового додатку, який би поєднував простоту використання та функціональність.

Основні завдання проекту були чітко окреслені в технічному завданні, що включало такі вимоги: розробка простого та інтуїтивно зрозумілого дизайну інтерфейсу; надання можливості створення та редагування гітарних табулатур; реалізація функції відтворення композицій за допомогою різних музичних інструментів; забезпечення можливості імпорту та експорту створених табулатур у файл. Моделювання роботи програмного додатку здійснювалося за допомогою нотації IDEF0 та діаграм варіантів використання (Use Case). У результаті було створено детальні діаграми, що допомогли візуалізувати та структурувати функціонал додатку.

Програма була розроблена на базі мови програмування Java із використанням бібліотеки JavaFX для створення зручного графічного інтерфейсу. Використання Java Sound API дало змогу реалізувати функціонал відтворення нот через MIDI синтезатор, що забезпечило високу якість звуку та підтримку різних музичних інструментів. У процесі розробки було заповнено три основних файли: Main.java для запуску програми та ініціалізації головного вікна, GuitarTabEditor.java як

контролер для обробки взаємодій користувача, та hello-view.fxml, що відповідає за графічний інтерфейс.

Результатом виконання цієї кваліфікаційної роботи став функціональний програмний додаток для створення, редагування та відтворення гітарних табулатур. Додаток відповідає всім вимогам, визначеним у технічному завданні, і надає користувачам зручний інструмент для роботи з музичними композиціями. Він поєднує в собі простоту використання та широкий функціонал, що робить його доступним та ефективним для музикантів будь-якого рівня.

Подальший розвиток цього проекту може включати додавання нових функцій, таких як підтримка інших музичних інструментів, розширення можливостей імпорту та експорту файлів, а також покращення інтерфейсу користувача для підвищення зручності використання. Це дозволить зробити додаток ще більш універсальним та корисним для користувачів, а також забезпечить його конкурентоспроможність на ринку програмного забезпечення для створення музики.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Довідник по java. *https://itproger.com/ua/spravka/java*.
2. Що таке табулатура та як в ній розібратись? *BUKI*.
3. Beatty J., Wiegers K. Software requirements. Microsoft Press, 2013.
4. Guitar pro - tab editor software for guitar, bass, drum, piano and more. *https://www.guitar-pro.com/*.
5. Java platform, standard edition (java SE) documentation. URL: https://docs.oracle.com/javase/8/docs/ (date of access: 25.04.2024).
6. Java tutorial | learn java programming. *https://www.geeksforgeeks.org/java/*. URL: https://www.geeksforgeeks.org/java/ (date of access: 25.04.2024).
7. Schwaber K. Agile project management with scrum. Microsoft Press, 2004.
8. Songsterr: guitar tabs with rhythm. *https://www.songsterr.com/*.
9. TuxGuitar. *https://www.tuxguitar.app/*.
10. Deitel, P., Deitel, H. (2015). Java: How to Program, 10th Edition. Prentice Hall.
11. Eckel, B. (2006). Thinking in Java, 4th Edition. Prentice Hall.
12. JavaFX Documentation. Available at: https://openjfx.io/
13. Java Sound API Documentation. Available at: https://docs.oracle.com/javase/8/docs/technotes/guides/sound/
14. UML Distilled: A Brief Guide to the Standard Object Modeling Language (3rd Edition) by Martin Fowler.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку

«Програмний додаток створення та редагування гітарних табулатур»

ПОГОДЖЕНО:

Старший викладач кафедри інформаційних технологій

_____ Кузнєцов Е .Г.

Студент групи ІТ-03

Маховик О. С.

Суми 2024

1. Призначення та мета програмного додатку "TabXpress"

1.1 Призначення програмного додатку

Програмний додаток "TabXpress" розроблено для гітаристів та музичних ентузіастів, які шукають зручний інструмент для створення, редагування та обміну табулатур (табів) для гітари в офлайн-режимі на своєму персональному комп'ютері.

1.2 Мета створення програмного додатку

Головна мета проекту - розробити програмний додаток "TabXpress" з метою спрощення процесу створення, редагування та обміну табулатур для гітари в офлайн-режимі. Проєкт спрямований на забезпечення гітаристів інструментом для творчості, обміну досвідом та поліпшення їхньої музичної практики без прив'язки до Інтернету.

1.3 Цільова аудиторія

Цей десктопний додаток призначений для гітаристів різного рівня підготовки, які шукають зручний інструмент для творчості з табулатурами в офлайн-режимі. Також включає музичних ентузіастів, які хочуть створювати та редагувати музичні нотації на своєму персональному комп'ютері.

2. Вимоги до проекту

2.1 Вимоги до проекту в цілому

2.1.1 Вимоги до структури й функціонування

Десктопний додаток "TabXpress" пропонує зручний та інтуїтивно зрозумілий користувацький інтерфейс, що надає легкий доступ до всіх функціональних можливостей через меню та інші елементи інтерфейсу. Функціонал додатка включає здатність створювати, відкривати та зберігати табулатури, редагування та форматування тексту і нотації табулатур, а також перегляд і відтворення створених табулатур. Усі дані зберігаються локально на комп'ютері користувача.

2.1.2 Вимоги до персоналу

Персонал, що працює з додатком, повинен мати базові навички користування комп'ютером та програмами.

2.1.3 Вимоги до збереження інформації

Усі дані, що створюються та редагуються в програмному додатку "TabXpress", повинні зберігатися локально на пристрої користувача. Для цього може використовуватися вбудована база даних або локальні файли.

2.1.4 Вимоги до розмежування доступу для програмного додатку "TabXpress"

Розроблюваний програмний додаток для створення та редагування гітарних табулатур передбачає встановлення обмежень доступу для різних категорій користувачів з метою забезпечення безпеки та конфіденційності інформації. Доступ до функціоналу розмежовується за наступними групами користувачів: адміністратор, звичайний користувач та авторизований користувач. Адміністратор має повний доступ до всіх функцій додатку, включаючи створення, редагування та видалення табулатур, а також керування користувачами. Звичайний користувач може створювати та редагувати табулатури, але не має доступу до адміністративних функцій. Авторизований користувач має розширені можливості

порівняно зі звичайним користувачем, такі як збереження персональних налаштувань та історії редагування.

2.2 Структура програмного додатку "TabXpress"

2.2.1 Загальна інформація про структуру програмного додатку

Структура програмного додатку для створення та редагування гітарних табулатур передбачає розподіл функцій та компонентів для забезпечення ефективності та зручності користування. Додаток складається з таких основних елементів:

1. Користувацький інтерфейс: Включає в себе елементи для відображення табулатур, інструменти редагування, панелі навігації та інші візуальні компоненти, що дозволяють користувачеві зручно працювати з додатком.
2. Меню і керуючі елементи: Містить команди для створення нових табулатур, відкриття існуючих файлів, збереження та експортування результатів.
3. Редактор табулатур: Це основна робоча область, де користувач може вводити та редагувати нотацію гітарних табулатур, використовуючи спеціальний інтерфейс з клавішами для нот та акордів.
4. Модуль відтворення: Надає можливість прослуховувати створені табулатури в реальному часі, щоб користувач міг перевірити правильність розміщення нот та звучання акустично.
5. Система зберігання даних: Використовується для збереження створених табулатур, історії редагування та інших важливих даних користувача.
6. Модуль імпорту та експорту: Надає можливість імпортувати та експортувати табулатури у різних форматах, щоб забезпечити сумісність з іншими програмами та сервісами.

2.2.2 Навігаційне меню

На верхній частині додатку розташоване навігаційне меню, яке фіксується для зручності користувача. Це меню містить посилання на всі доступні сторінки додатку.

2.2.3 Управління контентом

Управління контентом здійснюється через адміністративну панель. Дані для наповнення додатку зберігаються у базі даних. Адміністратор має повний доступ до редагування інформації.

2.2.4 Дизайн додатку "TabXpress"

Дизайн додатку виконаний у мінімалістичному та сучасному стилі. Використовуються корпоративні кольори: білий та помаранчевий. Шрифти мають комфортний розмір для читання. Розташування блоків та елементів є зручним та логічним, надаючи користувачеві приємний інтерфейс.

2.2.5 Система навігації (карта додатку)

Карта додатку зображена на рисунку А.1.

Рисунок А.1 – Система навігації

2.3 Вимоги до видів забезпечення

2.3.1 Вимоги до лінгвістичного забезпечення

Весь текст у програмному додатку має бути виконаний українською мовою та англійською мовами .

2.3.2 Вимоги до програмного забезпечення

Для забезпечення стабільної роботи додатку він повинен бути адаптований до роботи на операційних системах Windows 7 і вище.

2.4 Вимоги до функціонування системи

2.4.1 Потреби користувача

ID	Потреби користувача	Джерело
ТХ-01	Створення нової табулатури	Гітарист, Музикант
ТХ-02	Редагування існуючих табулатур	Гітарист, Музикант
ТХ-03	Перегляд та пошук доступних табулатур	Гітарист, Музикант
ТХ-04	Збереження та відновлення робочих сесій	Гітарист, Музикант
ТХ-05	Експорт та імпорт табулатур у різних форматах	Гітарист, Музикант
ТХ-06	Функція відтворення звуку для перевірки табуляцій	Гітарист, Музикант
ТХ-07	Можливість обміну табулатурами з іншими користувачами	Гітарист, Музикант
ТХ-08	Реєстрація та авторизація для збереження особистих даних	Гітарист, Музикант
ТХ-09	Зворотній зв'язок та підтримка від розробників	Усі користувачі
ТХ-10	Налаштування світлої/темної теми інтерфейсу	Усі користувачі
ТХ-11	Перегляд статистики користування та популярних табулатур	Адміністратор, Гітарист

2.4.2 Системні вимоги

Проаналізувавши потреби користувачів було визначено наступні вимоги:

- Створення табулатур в додатку:

Має бути реалізована функція пошуку для зручного створення табулатур.

- Редагування створених таблуатур:

Додаток повинен забезпечувати можливість користувачам створювати, редагувати та видаляти свої табулатури.

- Можливість імпорту/експорту табулатур:

Створені табулатури можна зберігати у вигляді текстового файлу та відкривати у програмному додатку

- Функціональна панель:

Наявність адміністраторської панелі для додавання, редагування та видалення інформації про табулатури.

3 Склад і зміст робіт зі створення додатку «TabExpress»

Детальний опис етапів створення додатку наведено в таблиці А.2.

Таблиця А.2 – Етапи створення додатку

№	Склад і зміст робіт	Строк розробки
1	Розробка дизайну інтерфейсу	10 днів
2	Створення архітектури додатку	15 днів
3	Розробка основної логіки додатку	20 днів

4	Розробка модулю пошуку та фільтрації табулатур	10 днів
5	Реалізація функцій реєстрації та авторизації	7 днів
6	Розробка модулю оформлення та редагування табулатур	15 днів
7	Створення адміністраторської панелі	10 днів
8	Beta-тестування та виправлення помилок	15 днів
9	Alpha-тестування та доробка функціоналу	20 днів
10	Розміщення на хостингу або публікація	5 днів
11	Тестування працездатності	5 днів
12	Написання супровідної документації	10 днів
13	Реліз десктопного додатку "TabXpress"	1 день
	Загальна тривалість робіт	148 днів

ДОДАТОК Б
Планування робіт

У світі сучасної музики гітарні табулатури набувають все більшої популярності, що робить програми для їх створення та редагування дуже потрібними. Дана дипломна робота спрямована на розробку програмного продукту "TabXpress", який стане зручним інструментом для гітаристів у записі, редагуванні та збереженні музичних нот. Цей додаток дасть змогу музикантам швидко та легко фіксувати свої ідеї та композиції, а також ділитися ними з іншими користувачами. З "TabXpress" гітаристи зможуть створювати та налаштовувати ноти для гітари, що значно полегшить їхній творчий процес та допоможе розвинути музичний талант.

Деталізація мети проекту методом SMART. Щоб проект був успішним та конкурентоспроможним треба на концептуальному етапі правильно визначити його мету за допомогою SMART-методу. Результати деталізації методом SMART розміщені у таблиці Б.1.

Таблиця Б.1 – Деталізація мети проекту методом SMART

Specific	Розробити табулатурний редактор, який надасть користувачам зручний інтерфейс для створення музичних нот.
Measurable	Забезпечити можливість створення та редагування табулатур з розширеними функціональними можливостями.
Achievable	Мета досяжна, оскільки маємо висококваліфікованих програмістів та розуміємо основні вимоги користувачів музичного ПЗ.
Relevant	Для спрощення та полегшення творчого процесу музикантів, особливо гітаристів, через доступний та ефективний редактор.
Time-framed	Завершення розробки до 1 червня 2024 р.

Планування змісту робіт. WBS (Work Breakdown Structure – Ієрархічна структура робіт) – це графічний вигляд елементів проекту, які згруповані ієрархією у єдине ціле з продуктом проекту. Структура декомпозиції робіт орієнтована на досконале виконання робіт по частинам і сама є ключовою частиною проекту, яка спрямована на організацію командної роботи. Елементами декомпозиції можуть бути продукти, дані та послуги. Більше того, WBS забезпечує необхідним каркасом для ретельної оцінки термінів та контролю та графіків роботи.

На найвищому (першому) рівні розміщений продукт проекту. Основні дії та заходи, що забезпечують досягнення мети проекту, зафіксовані на другому рівні декомпозиції. Декомпозиція робіт виконується до тих пір, поки вони не стануть елементарними (простими).

Елементарні роботи – це дії, які мають однозначний чіткий результат, на які призначена відповідальному одна конкретна особа, для якої можна обчислити витрати праці і тривалість виконання. На рисунку Б.1 представлено WBS з розробки програмного додатку «TabXpress».

Планування структури виконавців. Наступним етапом після декомпозиції процесів є розробка організаційної структури виконавців або OBS, яка визначається як графічна структура відображення учасників або відповідальних осіб, які беруть участь у реалізації проекту.

У ролі відповідальних осіб виступають співробітники, що відповідають за організацію і виконання елементарної роботи, що зазначена у WBS. Кожну елементарну роботи можна розглядати як окремий проект.

На рисунку Б.2 представлено організаційну структуру планування проекту. Список виконавців, що функціонують в проекті описано в таблиці Б.2

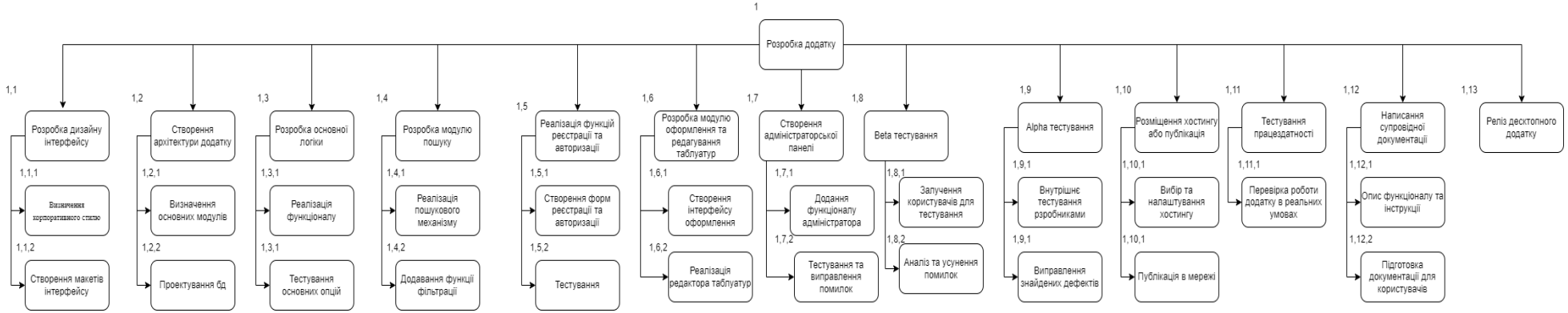

Рисунок Б.1 – WBS-структура робіт проекту

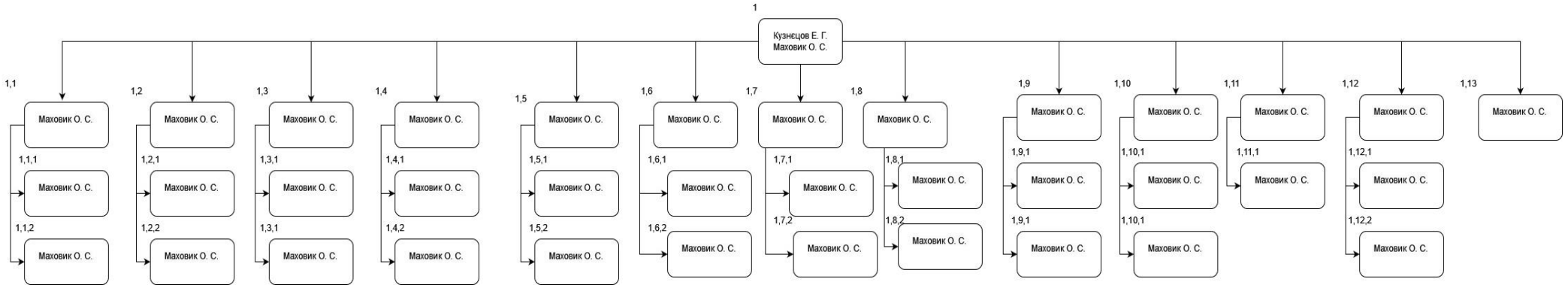

Рисунок Б.2 – OBS-структура робіт проекту

Таблиця Б.2 – Виконавці проєкту

Роль	Ім’я	Проектна роль
Розробник	Маховик О. С.	Виконує розробку
Проектувальник	Маховик О. С.	Виконує проектування бази даних та розробляє структуру додатку.
Тестувальник	Маховик О. С.	Відповідає за тестування функціоналу та дизайну додатку.
Керівник проекту	Кузнєцов Е.Г.	Формує завдання на розробку проекту.
Менеджер проекту	Маховик О. С.	Відповідає за виконання терміні. Виконує збір та аналіз даних.

Діаграма Ганта. Побудова календарного графіку (діаграми Ганта) є одним з важливих етапів планування проекту, що виглядає як розклад виконання робіт з реальним розподілом дат. Завдяки йому можна отримати достовірне уявлення про тривалість процесів з обмеженнями у ресурсах, урахуванням вихідних днів та свят.

Календарний графік проекту представлено на рисунку Б.3

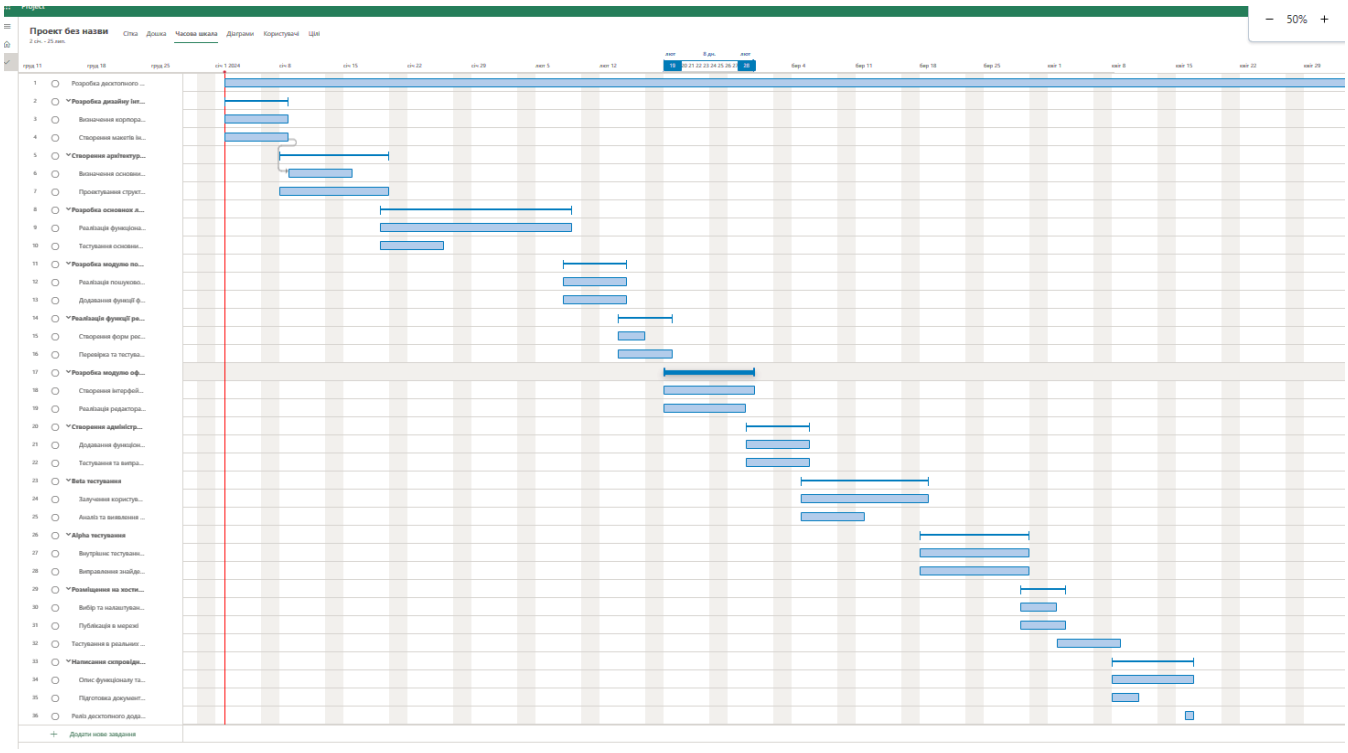

Рисунок Б.3 – Календарний графік проекту

Управління ризиками проекту. Під час управління ризиками проекту необхідно провести докладну оцінку ризиків, визначивши ті з них, які потребують негайного врегулювання. Реакція на ризик визначається його важливістю: деякі ризики потребують негайних заходів, тоді як інші можуть бути вирішені згідно їхнього потенційного впливу. Після цього проводиться кількісне оцінювання ризиків, що може виконуватися паралельно з якісною оцінкою або окремо, залежно від потреб проекту. У таблиці Б.3 описано ризики та стратегії реагування на кожен із них.

Таблиця Б.3 Ризики та стратегії реагування

D	Статус	Опис	Ймовірність виникнення	Вплив	Ранг	План А (заходи запобігання виникненню ризику)	Типи стратегії реагування	План Б (заходи усунення наслідків ризику)
1	Відкритий	Нечітке завдання на розробку	0,4	0,5	0,2	1. Грамотно поставити технічне завдання проекту. 2. Звітування перед замовником на різних етапах роботи. 3. Уникати нечітких, та невимірюваних характеристик продукту.	Попередження	Якщо є розбіжності між характеристиками продукту та ТЗ - виконати виправлення.
2	Відкритий	Низька кваліфіка-ція розробників	0,4	0,5	0,2	Провести ретельний відбір серед кваліфокаваних претендентів за допомогою рекрутерів.	Попередження	Якщо робітник або ціла команда виявилися недостатньо компетентними – передати проєкт іншій команді або знайти нового робітника за допомогою рекрутера.
3	Відкритий	Зміни у ТЗ	0,3	0,2	0,06	Провести аналіз предметної області. Обговорення з командою та замовником. Визначити нове ТЗ.	Ухилення	Повне перепланування

Продовження таблиці Б.3

4	Відкритий	Неоптимальний розподіл часу	0,3	0,2	0,06	Розрахувати час виконання замовлення із запасом.	Попередження	Повне перепланування з урахування усіх обтяжуючих обставин.
5	Відкритий	Повітряні тривоги	0,5	0,9	0,45	Перемістити офіс подалі від бойових дій. Допомогти учасникам проекту покинути обстрілювану територію.	Ухилення	Перенести строки виконання робіт. Перепланування.
6	Відкритий	Перебої електропостачання	0,7	0,7	0,49	1. Виконувати роботу на ноутбуках, або інших пристроях з автономними джерелами живлення. 2. Закупити переносні генератори та потужні павербанків 3. Закласти в календарний план додаткові резерви. 4. Мати резервне джерело мобільного інтернету. 5. Заряджати робочі інструменти при кожній нагоді.	Ухилення	Переміщення офісу або робітників у населений пункт з стабільної енергосистемою

ДОДАТОК В

Лістинг коду програмного додатку

Файл: Main.java

```
package com.example.demo1;

import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
import javafx.stage.Stage;

import java.io.IOException;

public class Main extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(Main.class.getResource("hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), 800, 600);
        stage.setTitle("Guitar Tab Editor");
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }
}
```

Файл: GuitarTabEditor.java

```
package com.example.demo1;

import javafx.application.Platform;
import javafx.fxml.FXML;
import javafx.scene.Scene;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.input.KeyEvent;
import javafx.scene.layout.GridPane;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

import javax.sound.midi.*;
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class GuitarTabEditor {

    @FXML
    private GridPane tabGrid;
    @FXML
    private TextField bpmField;
    @FXML
    private ComboBox<String> instrumentComboBox;

    private Stage stage;
    private List<List<TextField>> tabFields;
    private int currentCols = 12; // поточна кількість стовпців
    private int bpm = 120; // Значення BPM за замовчуванням

    private int[] instruments = {25, 0, 27}; // 25 - акустична гітара, 0 - піаніно, 27 -
```

електрогітара

```
        private int currentInstrument = instruments[0];

        public void setStage(Stage stage) {
            this.stage = stage;
        }

        @FXML
        public void initialize() {
            tabFields = new ArrayList<>();
            initializeTabGrid();

            // Заповнення ComboBox інструментами
            instrumentComboBox.getItems().addAll("Acoustic Guitar", "Piano", "Electric Guitar");
            instrumentComboBox.setValue("Acoustic Guitar");
            instrumentComboBox.setOnAction(e -> updateInstrument());
        }

        private void initializeTabGrid() {
            tabGrid.getChildren().clear();
            String[] stringNames = {"e", "B", "G", "D", "A", "E"};

            // Додавання назв струн зліва
            for (int i = 0; i < 6; i++) {
                Label label = new Label(stringNames[i]);
                label.setStyle("-fx-text-fill: #2b5797;");
                tabGrid.add(label, 0, i);
            }

            // Додавання текстових полів для табулатури
            tabFields.clear();
            for (int i = 0; i < 6; i++) {
                List<TextField> row = new ArrayList<>();
                for (int j = 1; j <= currentCols; j++) {
                    TextField textField = createNoteField();
```

```
            tabGrid.add(textField, j, i);
            row.add(textField);
        }
        tabFields.add(row);
    }
}

private TextField createNoteField() {
    TextField textField = new TextField();
    textField.setPrefWidth(30);
    textField.addEventFilter(KeyEvent.KEY_PRESSED, event -> {
        switch (event.getCode()) {
            case RIGHT:
                moveFocus(textField, 1, 0);
                event.consume();
                break;
            case LEFT:
                moveFocus(textField, -1, 0);
                event.consume();
                break;
            case UP:
                moveFocus(textField, 0, -1);
                event.consume();
                break;
            case DOWN:
                moveFocus(textField, 0, 1);
                event.consume();
                break;
            default:
                break;
        }
    });
    return textField;
}
```

```
    private void moveFocus(TextField currentField, int colShift, int rowShift) {
        for (int row = 0; row < tabFields.size(); row++) {
            for (int col = 0; col < tabFields.get(row).size(); col++) {
                if (tabFields.get(row).get(col) == currentField) {
                    int newRow = Math.min(Math.max(row + rowShift, 0), tabFields.size() - 1);
                    int newCol = Math.min(Math.max(col + colShift, 0), tabFields.get(row).size() -
1);
                    tabFields.get(newRow).get(newCol).requestFocus();
                    return;
                }
            }
        }
    }

    @FXML
    private void newTab() {
        tabFields.clear();
        currentCols = 12; // скидання до початкової кількості стовпців
        initializeTabGrid();
    }

    @FXML
    private void openTab() {
        FileChooser fileChooser = new FileChooser();
        fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Text Files",
"*.txt"));
        File file = fileChooser.showOpenDialog(stage);
        if (file != null) {
            try (BufferedReader reader = new BufferedReader(new FileReader(file))) {
                String line = reader.readLine();
                if (line != null) {
                    bpm = Integer.parseInt(line); // Читання BPM з першого рядка файлу
                    bpmField.setText(String.valueOf(bpm));
                }
                line = reader.readLine();
```

```
                    if (line != null) {
                        currentCols = Integer.parseInt(line); // Читання кількості стовпців з другого
рядка файлу
                    }
                    initializeTabGrid();
                    for (int i = 0; i < 6; i++) {
                        line = reader.readLine();
                        if (line != null) {
                            String[] notes = line.split(" ");
                            for (int col = 0; col < notes.length; col++) {
                                tabFields.get(i).get(col).setText(notes[col]);
                            }
                        }
                    }
                } catch (IOException e) {
                    showError("Error", "Could not load the file.");
                    e.printStackTrace();
                }
            }
        }

        @FXML
        private void saveTab() {
            FileChooser fileChooser = new FileChooser();
            fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter("Text Files",
"*.txt"));
            File file = fileChooser.showSaveDialog(stage);
            if (file != null) {
                try (FileWriter writer = new FileWriter(file)) {
                    writer.write(bpmField.getText() + "\n"); // Запис BPM у файл
                    writer.write(currentCols + "\n"); // Запис кількості стовпців у файл
                    for (int i = 0; i < 6; i++) {
                        for (int j = 0; j < currentCols; j++) {
                            writer.write(tabFields.get(i).get(j).getText() + " ");
                        }
```

```
                    writer.write("\n");
                }
            } catch (IOException e) {
                showError("Error", "Could not save the file.");
                e.printStackTrace();
            }
        }
    }

    @FXML
    private void playTab() {
        try {
            bpm = Integer.parseInt(bpmField.getText());
            int duration = 60000 / bpm; // Обчислення тривалості кожної ноти в мілісекундах

            Synthesizer synthesizer = MidiSystem.getSynthesizer();
            synthesizer.open();
            MidiChannel[] channels = synthesizer.getChannels();

            synthesizer.loadInstrument(synthesizer.getDefaultSoundbank().getInstrument(new Patch(0,
currentInstrument)));
            channels[0].programChange(currentInstrument);

            for (int j = 0; j < currentCols; j++) {
                ArrayList<Integer> notesToPlay = new ArrayList<>();
                for (int i = 0; i < 6; i++) {
                    String text = tabFields.get(i).get(j).getText();
                    if (!text.isEmpty()) {
                        int note = Integer.parseInt(text);
                        notesToPlay.add(note + getBaseNoteForString(i));
                        channels[0].noteOn(note + getBaseNoteForString(i), 600);
                    }
                }
                highlightColumn(j + 1, true);
                Thread.sleep(duration); // Пауза між стовпцями з врахуванням BPM
```

```
            highlightColumn(j + 1, false);
            for (int note : notesToPlay) {
                channels[0].noteOff(note);
            }
        }
        synthesizer.close();
    } catch (Exception e) {
        showError("Error", "An error occurred during playback.");
        e.printStackTrace();
    }
}

@FXML
private void addColumn() {
    currentCols++;
    for (int i = 0; i < 6; i++) {
        TextField textField = createNoteField();
        tabGrid.add(textField, currentCols, i);
        tabFields.get(i).add(textField);
    }
}

@FXML
private void exitApp() {
    Platform.exit();
}

private int getBaseNoteForString(int stringIndex) {
    switch (stringIndex) {
        case 0: return 64; // Нота E2
        case 1: return 59; // Нота A2
        case 2: return 55; // Нота D3
        case 3: return 50; // Нота G3
        case 4: return 45; // Нота B3
        case 5: return 40; // Нота E4
```

```
            default: return 40;
        }
    }

    private void highlightColumn(int col, boolean highlight) {
        for (int i = 0; i < 6; i++) {
            TextField field = tabFields.get(i).get(col - 1);
            if (highlight) {
                field.setStyle("-fx-background-color: yellow;");
            } else {
                field.setStyle("");
            }
        }
    }

    private void updateInstrument() {
        String selectedInstrument = instrumentComboBox.getValue();
        switch (selectedInstrument) {
            case "Acoustic Guitar":
                currentInstrument = instruments[0];
                break;
            case "Piano":
                currentInstrument = instruments[1];
                break;
            case "Electric Guitar":
                currentInstrument = instruments[2];
                break;
        }
    }

    private void showError(String title, String message) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle(title);
        alert.setHeaderText(null);
        alert.setContentText(message);
```

```
        alert.showAndWait();
    }
}
```

Файл: Hello-view.fxm

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.Menu?>
<?import javafx.scene.control.MenuBar?>
<?import javafx.scene.control.MenuItem?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.BorderPane?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>

<BorderPane xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.demo1.GuitarTabEditor">
  <top>
    <MenuBar>
      <Menu text="File">
        <MenuItem text="New" onAction="#newTab"/>
        <MenuItem text="Open" onAction="#openTab"/>
        <MenuItem text="Save" onAction="#saveTab"/>
        <MenuItem text="Exit" onAction="#exitApp"/>
      </Menu>
    </MenuBar>
    <HBox spacing="10.0" alignment="CENTER" style="-fx-padding: 10;">
      <Button id="newTab" text="New" onAction="#newTab" styleClass="button" graphicTextGap="10.0" />
```

```
          <Button id="openTab" text="Open" onAction="#openTab" styleClass="button" graphicTextGap="10.0"
/>
          <Button id="saveTab" text="Save" onAction="#saveTab" styleClass="button" graphicTextGap="10.0"
/>
          <Button id="playTab" text="Play" onAction="#playTab" styleClass="button" graphicTextGap="10.0"
/>
          <Button id="addColumn" text="Add Column" onAction="#addColumn" styleClass="button"
graphicTextGap="10.0" />
          <Label text="BPM:" styleClass="label"/>
          <TextField fx:id="bpmField" text="120" styleClass="text-field" style="-fx-pref-width: 50;"/>
          <Label text="Instrument:" styleClass="label"/>
          <ComboBox fx:id="instrumentComboBox" styleClass="combo-box"/>
          <Button text="Exit" onAction="#exitApp" styleClass="button" graphicTextGap="10.0" />
        </HBox>
      </top>
      <center>
        <ScrollPane style="-fx-padding: 10;">
          <VBox spacing="10.0" alignment="CENTER">
            <GridPane fx:id="tabGrid" hgap="10.0" vgap="10.0" style="-fx-padding: 10;">
              <!-- Табулатурні поля будуть додаватись тут -->
              <Label text="e" GridPane.rowIndex="0" GridPane.columnIndex="0" styleClass="label"/>
              <Label text="B" GridPane.rowIndex="1" GridPane.columnIndex="0" styleClass="label"/>
              <Label text="G" GridPane.rowIndex="2" GridPane.columnIndex="0" styleClass="label"/>
              <Label text="D" GridPane.rowIndex="3" GridPane.columnIndex="0" styleClass="label"/>
              <Label text="A" GridPane.rowIndex="4" GridPane.columnIndex="0" styleClass="label"/>
              <Label text="E" GridPane.rowIndex="5" GridPane.columnIndex="0" styleClass="label"/>
            </GridPane>
          </VBox>
        </ScrollPane>
      </center>
    </BorderPane>l
```