НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва інституту/факультету)

кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва кафедри)

«До захисту допущено»
В.о. завідувачки кафедри БМК

_____ Світлана АЛХІМОВА
(підпис) (ініціали, ПРІЗВИЩЕ

"___" __*червня*__ 2025р.

Дипломна робота

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Комп'ютерні технології в біології та медицині»
зі спеціальності *122 «Комп'ютерні науки»*

на тему**:** **Програмний застосунок для визначення предиктора раптової серцевої смерті за електрокардіограмою (ЕКГ)**

Виконав: студент **IV** курсу, групи _БС-12__

ІСЬКОВ ОЛЕКСАНДР ОЛЕКСАНДРОВИЧ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник: *Проф. каф. біомедичної кібернетики (БМК) д.т.н., професор Файнзільберг Леонід Соломонович*

(посада, науковий ступінь, вчене звання, прізвище, ім'я, по ініціали) (підпис)

Консультант з розділів дипломної роботи:

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент: *професор каф. біомедичної інженерії, д.м.н. Максименко Віталій Борисович*

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інститут (факультет) БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ
(повна назва)

Кафедра БІОМЕДИЧНОЇ КІБЕРНЕТИКИ
(повна назва)

Рівень вищої освіти – перший (бакалаврський)
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Комп'ютерні технології в біології та медицині»
(код і назва)

ЗАТВЕРДЖУЮ
В.о. завідувачки кафедри БМК
_____ Світлана АЛХІМОВА
«_ ___ »__*квітень*___2025 р.

ЗАВДАННЯ
на дипломну роботу студенту

ІСЬКОВУ ОЛЕКСАНДРУ ОЛЕКСАНДРОВИЧУ
(прізвище, ім'я, по батькові)

1. Тема роботи **Програмний застосунок для визначення предиктора раптової серцевої смерті за електрокардіограмою (ЕКГ)**

керівник роботи
Файнзільберг Леонід Соломонович, д.т.н., професор
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_26_»_*травня*_ 2025 р. №***1758-с***_

2. Термін подання студентом роботи ***22-23 травня 2025 року***

3. Вихідні дані: синтетичні ЕКГ-сигнали, інструментарій для розробки програмного застосунку.

4. Зміст роботи: огляд існуючих неінвазивних предикторів раптової серцевої смерті, імплементація математичного алгоритму виявення предиктора раптової серцевої смерті, побудова архітектури програмного застосунку, програмна реалізація застосунку для визначення предиктора раптової серцевої смерті.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація для захисту дипломної роботи.

6. Консультанти розділів роботи[1*]

[1*] Якщо визначені консультанти. Консультантом не може бути зазначено наукового керівника магістерської дисертації.

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Дипломної роботи			

7. Дата видачі завдання ***06 квітня 2025р.***

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримати завдання за темою ДР на практику	До 24.12.2024р.	***виконано***
2	Переддипломна практика. Виконання практичної частини ДР та додаткових розділів	1-4 тиждень за графіком практики	***виконано***
3	Виконання основних розділів ДР (Огляд неінвазивних предикторів раптової серцевої смерті, Засоби реалізації програмного застосунку, Програмна реалізація та методика роботи програмного застосунку)	Протягом усієї практики	***виконано***
4	Перевірка ДР науковим керівником	до 20.05.2025	***виконано***
5	Подання в електронному вигляді ДР на перевірку нормоконтролера та плагіат (StrikePlagiarism.com). Доопрацювання ДР	Не пізніше 23.05.2025	***виконано***
6	Отримати відгук від керівника ДР	Не пізніше 05.06.2025	***виконано***
7	Надати пакету документів по ДР та супровідних до неї документів на засідання кафедри		***виконано***
8	Подання ДР рецензенту. Отримання рецензії.	09.06 – 13.06.2025	***виконано***
9	Захист ДР в ЕК	*16.06 - -21.06.2025*	

Студент _____ **Олександ ІСЬКОВ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Керівник ДР _____ **Леонід ФАЙНЗІЛЬБЕРГ**
(підпис) (ім'я, ПРІЗВИЩЕ)

Нормоконтролер _____ **Галина КОРНІЄНКО**
(підпис) (ім'я, ПРІЗВИЩЕ)

АНОТАЦІЯ

Дипломна робота за темою «Програмний застосунок для визначення предиктора раптової серцевої смерті за електрокардіограмою (ЕКГ)» виконана студентом кафедри біомедичної кібернетики ФБМІ Іськовим Олександром Олександровичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині» та складається зі: вступу; 3 розділів (огляд неінвазивних предикторів раптової серцевої смерті, засоби реалізації програмного застосунку, програмна реалізація та методика роботи програмного застосунку), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 55 джерел, та додатків. Загальний обсяг роботи 90 сторінок.

Актуальність теми. Актуальність теми дипломної роботи пов'язана із існуючою глобальною проблемою раптової серцевої смерті та необхідністю створення доступного програмного застосунку для автоматизованого виявлення чутливого неінвазивного предиктора раптової серцевої смерті, що дозволить покращити ранню діагностику кардіологічних ризиків та знизити трудомісткість ручного аналізу ЕКГ сигналів у клінічній та амбулаторній практиці.

Мета і завдання роботи.

Метою роботи є підвищення достовірності виявлення ризику раптової серцевої смерті шляхом створення програмного застосунку для аналізу електрокардіографічних сигналів.

Її досягнення передбачає вирішення наступних завдань:

1. Аналітичний огляд літератури щодо електрокардіографічних предикторів раптової серцевої смерті.
2. Здійснення оцінки доступності та репрезентативності відкритих наборів електрокардіографічних даних та за потреби, обґрунтування доцільності синтетичного моделювання сигналів, визначивши

вимоги до відповідного генератору.

3. Виконання дослідження та порівняння існуючих алгоритмів для детекції обраного предиктора, обрання алгоритму, що найкраще відповідає поставленим вимогам обчислювальної складності та точності детекції.
4. Формулювання вимог до програмного застосунку.
5. Здійснення проєктування високорівневої архітектури застосунку, опис протоколу взаємодії між модулями генерації, аналізу сигналів, зберігання даних та користувацького інтерфейсу.
6. Реалізація прототипу, що інтегрує обраний алгоритм детекції з визначеним набором технологій.
7. Проведення тестування програмного застосунку для визначення точності й продуктивності алгоритму на різних конфігураціях вхідного сигналу.

Використані методи. Мова програмування Java, фреймворк об'єктно-реляційного відображення Hibernate, графічний інструментарій JavaFX, система управління базами даних PostgreSQL, інструмент з автоматизації збірки та управління залежностями Maven.

Отримані результати. Програмний застосунок для визначення предиктора раптової серцевої смерті, альтернації зубця Т за електрокардіограмою.

Ключові слова. Раптова серцева смерть, Java, програмний застосунок, альтернація зубця Т, предиктор, ЕКГ.

Бібліографічний опис ДР

Іськов О. О. [Програмний застосунок для визначення предиктора раптової серцевої смерті за електрокардіограмою (ЕКГ)] : дипломна роб. бакалавра : 122 Комп'ютері науки / Іськов Олександр Олександрович. – Київ, 2025. – 90 с

ANNOTATION

The bachelor's thesis on the topic "Software Application for Determining a Predictor of Sudden Cardiac Death from Electrocardiogram (ECG)" was performed by the student of the Department of Biomedical Cybernetics, FBME, Iskov Oleksandr Oleksandrovych, specialty 122 "Computer Science" under the educational and professional program "Computer Technologies in Biology and Medicine," and consists of: introduction; 3 chapters (Review of Non-Invasive Predictors of Sudden Cardiac Death; Means of Implementing the Software Application; Software Implementation and Methodology of the Software Application); conclusions to each of these chapters; general conclusions; list of references, which includes 55 sources; and appendices. Total volume of the work is 90 pages.

Actuality of topic. The relevance of the diploma thesis topic is related to the existing global problem of sudden cardiac death and the necessity of creating an accessible software application for the automated detection of a sensitive non-invasive predictor of sudden cardiac death, which will improve early diagnosis of cardiac risks and reduce the labor intensity of manual ECG signal analysis in clinical and outpatient practice.

The purpose and tasks of the research. The aim of the work is to increase the reliability of sudden cardiac death risk detection by creating a software application for the analysis of electrocardiographic signals. Achieving this aim involves solving the following tasks:

1. Analytical review of the literature on electrocardiographic predictors of sudden cardiac death.
2. Assessment of the availability and representativeness of open ECG datasets and, if necessary, justification of the feasibility of synthetic signal modeling by defining requirements for the corresponding generator.
3. Research and comparison of existing algorithms for detection of the chosen predictor; selection of the algorithm that best meets the specified

requirements for computational complexity and detection accuracy.

4. Formulation of requirements for the software application.
5. Design of the high-level architecture of the application; description of the interaction protocol between the signal generation, analysis, data storage modules, and the user interface.
6. Implementation of a prototype integrating the selected detection algorithm with the chosen set of technologies.
7. Testing of the software application to determine the accuracy and performance of the algorithm on various input signal configurations.

Methods Used. Java programming language; Hibernate Object-Relational Mapping framework; JavaFX graphical toolkit; PostgreSQL database management system; Maven build automation and dependency management tool.

Results obtained. A software application for determining the predictor of sudden cardiac death—T-wave alternans—from the electrocardiogram.

Keywords. Sudden cardiac death, Java, software application, T-wave alternans, predictor, ECG.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База даних, це структурована колекція даних, яка зберігається та управляється комп'ютерною системою.

ДР — дипломна робота.

ЕКГ — електрокардіограма.

РСС — раптова серцева смерть.

API — *Application Programming Interface*, інтерфейс прикладного програмування.

CSS — *Cascading Style Sheets*, каскадні таблиці стилів.

CRUD — *Create, Read, Update, Delete* — базові операції над даними.

DAO — *Data Access Object*, об’єкт доступу до даних.

GUI — *Graphical User Interface*, графічний інтерфейс користувача.

HTML — *HyperText Markup Language*, мова гіпертекстової розмітки.

HRV — *Heart Rate Variability*, варіабельність серцевого ритму.

JDBC — *Java Database Connectivity*, інтерфейс з’єднання Java із СУБД.

JIT — *Just-In-Time Compilation*, динамічна компіляція байт-коду.

JPA — *Jakarta Persistence API*, стандарт об’єктно-реляційного відображення.

JVM — *Java Virtual Machine*, віртуальна машина Java.

Java SE — *Java Standard Edition*, стандартна редакція Java.

Java FX — платформа Java для побудови графічних застосунків.

MMA — *Modified Moving Average*, модифікований алгоритм ковзного середнього.

Maven — система керування збіркою проектів Java.

ORM — *Object-Relational Mapping*, об’єктно-реляційне відображення.

PGSQL (PostgreSQL) — реляційна система керування базами даних PostgreSQL.

QT — інтервал QT електрокардіограми.

RR — інтервал RR (час між послідовними R-піками ЕКГ).

RMSSD — *Root Mean Square of Successive Differences*, корінь середнього квадрата різниць послідовних NN-інтервалів.

SDNN — *Standard Deviation of NN-intervals*, середньоквадратичне відхилення NN-інтервалів.

SQL — *Structured Query Language*, мова структурованих запитів.

SSL — *Secure Sockets Layer*, протокол захищеного з’єднання.

TWA — *T-Wave Alternans*, альтернація зубця Т.

UML — *Unified Modeling Language*, універсальна мова моделювання.

XML — *eXtensible Markup Language*, розширювана мова розмітки.

ВСТУП

Актуальність дипломної роботи пов’язана із існуючою глобальною проблемою раптової серцевої смерті та необхідністю створення доступного програмного застосунку для автоматизованого виявлення чутливого неінвазивного предиктора раптової серцевої смерті, що дозволить покращити ранню діагностику кардіологічних ризиків та знизити трудомісткість ручного аналізу ЕКГ сигналів у клінічній та амбулаторній практиці.

Створення програмного застосунку може допомогти оперативно та достовірно виявляти предиктор раптової серцевої смерті, надаючи лікарям інтерактивні візуалізації й висновки, що прискорює клінічне ухвалення рішень, розширює можливості амбулаторного моніторингу та знижує витрати на кардіодіагностику.

Мета і завдання роботи

Метою роботи є підвищення достовірності виявлення ризику раптової серцевої смерті шляхом створення програмного застосунку для аналізу електрокардіографічних сигналів.

Її досягнення передбачає вирішення наступних завдань:

1. Аналітичний огляд літератури щодо електрокардіографічних предикторів раптової серцевої смерті.
2. Здійснення оцінки доступності та репрезентативності відкритих наборів електрокардіографічних даних та за потреби, обґрунтувати доцільність синтетичного моделювання сигналів, визначивши вимоги до відповідного генератору.
3. Виконання дослідження та порівняння існуючих алгоритмів для детекції обраного предиктора, обрання алгориту, що найкраще відповідає поставленим вимогам обчислювальної складності та точності детекції.
4. Формулювання вимог до програмного застосунку.

5. Здіснення проєктування високорівневої архітектури застосунку, опис протокол взаємодії між модулями генерації, аналізу сигналів, зберігання даних та користувацького інтерфейсу.

6. Реалізація прототипу, що інтегрує обраний алгоритм детекції з визначеним набором технологій.

7. Проведення тестування програмного застосунку для визначення точності й продуктивності алгоритму на різних конфігураціях вхідного сигналу.

Використані методи. Мова програмування Java, фреймворк об'єктно-реляційного відображення Hibernate, графічний інструментарій JavaFX, система управління базами даних PostgreSQL, інструмент з автоматизації збірки та управління залежностями Maven.

Отримані результати. Програмний застосунок для визначення предиктора раптової серцевої смерті, альтернації зубця Т за електрокардіограмою.

Структура роботи

Дипломна робота за темою «Програмний застосунок для визначення предиктора раптової серцевої смерті за електрокардіогра́мою (ЕКГ)» виконана студентом Іськовим Олександром Олександровичем зі спеціальності 122 «Комп'ютерні науки» за освітньо-професійною програмою «Комп'ютерні технології в біології та медицині», побудована за класичним типом та викладена на 90 сторінках машинописного тексту. Вона складається з: вступу; 3 розділів (огляд неінвазивних предикторів раптової серцевої смерті, засоби реалізації програмного застосунку, програмна реалізація та методика роботи програмного застосунку), висновків до кожного з цих розділів; загальних висновків; списку використаних джерел, який налічує 55 джерел та додатків (22 – на кирилиці, 33 – на латиниці). В роботі представлено 43 рисунки і 0 таблиць

РОЗДІЛ 1
ОГЛЯД НЕІНВАЗИВНИХ ПРЕДИКТОРІВ РАПТОВОЇ СЕРЦЕВОЇ СМЕРТІ

В розділі подано узагальнений огляд літератури, присвяченої раптовій серцевій смерті (РСС) та її електрокардіографічним предикторам. Спершу уточнюються сучасні дефініції й класифікації РСС, відображаючи еволюцію критеріїв раптовості та кардіального генезу. Далі розглядається епідеміологія явища, окреслюються провідні етіологічні групи й віково-гендерні тенденції.

Наступний блок присвячено патофізіологічним механізмам, що лежать в основі фатальних аритмій, із наголосом на взаємодії структурних і електрофізіологічних чинників міокарда. Секція про електрокардіографічні характеристики детально описує ключові неінвазивні маркери ризику— подовження та дисперсію інтервалу QT, синдром Бругада та альтернацію зубця T — їхню фізіологічну природу, прояви на ЕКГ і прогностичну вагу. Далі порівнюються сильні й слабкі сторони кожного з предикторів, що обґрунтовує вибір альтернації зубця T як основного об’єкта подальшої комп’ютерної імплементації.

Завершує розділ огляд існуючих методів математичного моделювання T-альтернації, від класичних біофізичних моделей до спрощених генеративних підходів, які забезпечують синтетичні датасети для тестування алгоритмів автоматизованого аналізу.

1.1 Визначення та класифікація раптової серцевої смерті

Ключовий термін нашої роботи, «Раптова серцева смерть» брав участь у практичному застосуванні впродовж досить великої кількості років під різноманітними інтерпритаціями самого визначення. Інтерпретації ці були пов’язані з розбіжним поглядом на момент, коли серцева смерть починатиме

називатись раптовою, та кардіальне походження самої смерті.

Наразі, головними підходами до формулювання визначення та класифікації серцевої смерті, як раптової, є нетравматичний характер випадку, неочікуваність та миттєвість [1].

З метою розділення коронарного та некоронарного генезу РСС, раптовою смертю вважали випадок з клінічними проявами до 24 годин [1], однак в подальшому, в формулюванні визначення, цей строк зменшився до 1 години. В окремих інтерпритаціях, раптовість розглядають як смерть після 2-3 годин з моменту проявів відповідних клінічних проявів [2].

В контексті данної роботи, ми використовуватимемо визначення, запропоноване рекомендаціями Європейського кардіологічного товариства РСС, де раптову серцеву смерть визначають як «природну смерть внаслідок серцевих причин, якій передує раптова втрата свідомості протягом 1 години після початку гострих симптомів; можливое діагностоване раніше захворювання серця, але час і спосіб настання смерті несподівані» [1].

Згідно з науково-виробничим виданням асоціації кардіологів України “СЕРЦЕВО-СУДИННІ ЗАХВОРЮВАННЯ, Класифікація, стандарти діагностики та лікування”, раптову серцеву смерть визначають як смерть, що настала протягом попередніх 24 годин після появи перших симптомів заховрювання або суттєвого погіршення стану хворого на тлі стабільного хронічного перебігу захворювання [2], та поділяють поняття раптової серцевої смерті з відновленням серцевої діяльності (фібриляція шлуночків, асистолія, електромеханічна дисоціація) та незворотньої РСС (зупинка серця), коли смерть настала більше ніж через 1 годину після появи чи посилення сиптомів захворювання [2].

1.2 Епідеміологія раптової серцевої смерті

Європейське кардіологічне товариство визнає чотири групи факторів, які можуть вплинути на раптову серцеву смерть, серед яких, найбільш

численною групою є структурна хвороба серця з систолічною дисфункцією лівого шлуночка, друга група узагальнює субклінічні ураження міокарда, такі, як гіпертрофічну кардіоміопатію та інші [2] (рис. 1.1). До третьої групи відносяться минущі та ятрогенні фактори, а саме, аритмогенні ефекти, електролітний та вегетативний дисбаланс. До четвертої групи належать електричні хвороби, такі як синдроми QT, синдром Бругада, ідіопатична фібриляція шлуночків та синдром Волфа-Паркінсона-Уайта [3].

Рисунок 1.1 — Спектр етіології раптової серцевої смерті

Найпоширенішою клінічною ознакою, пов’язаною з раптовою серцевою смертю, є ішемічна хвороба серця, і приблизно 80% раптових серцевих смертей пов’язані з цією хворобою [3]. Інші 10–15% зустрічаються у пацієнтів із такими кардіоміопатіями, як гіпертрофічна кардіоміопатія, дилатаційна кардіоміопатія, аритмогенна дисплазія правого шлуночка та інфільтративні захворювання міокарда (саркоїд, амілоїдоз) [5]. Враховуючи

такий розповсюджений характер ішемічної хвороби серця, ймовірно, що цей стан значно збігається з кардіоміопатіями, тобто є пацієнти, які мають обидва захворювання, і ймовірно, що обидві етіології сприяють ризику раптової серцевої смерті [3]. Решта 5–10% складають або структурно аномальні вроджені захворювання серця, або пацієнти зі структурно нормальним, але електрично аномальним серцем. Цілком імовірно, що у певної частини цих пацієнтів, з часом розвинеться структурна хвороба серця, якщо вони переживуть раптову зупинку серця [5].

Говорячи про вікові та гендерні тенденції, існує два чітко встановлених піки вікової поширеності раптової серцевої смерті. Один у дитинстві, який ми називаємо синдромом раптової дитячої смерті, інший у геріатричній віковій групі, у віці від 75 до 85 років [2]. В дослідженні на основі свідоцтва про смерть у великій спільноті США [4], автори визначали щорічну частоту раптової серцевої смерті у загальній популяції, проводячи проспективну оцінку РРС серед усіх жителів округу Малтнома, штат Орегон, з населенням в 660486 осіб (рис. 1.2). В результаті дослідженнь, було виявлено, що раптова серцева смерть в цьому регіоні становить 5,6% щорічної смертності. Досвід штату Орегон показує, що на щорічній основі, жінки становлять 40% від усіх випадків раптової серцевої смерті (рис. 1.3) [4].

Рисунок 1.2 — Щорічна частота раптової серцевої смерті серед жителів округу Малтнома, штат Орегон (660 486 осіб) за віком

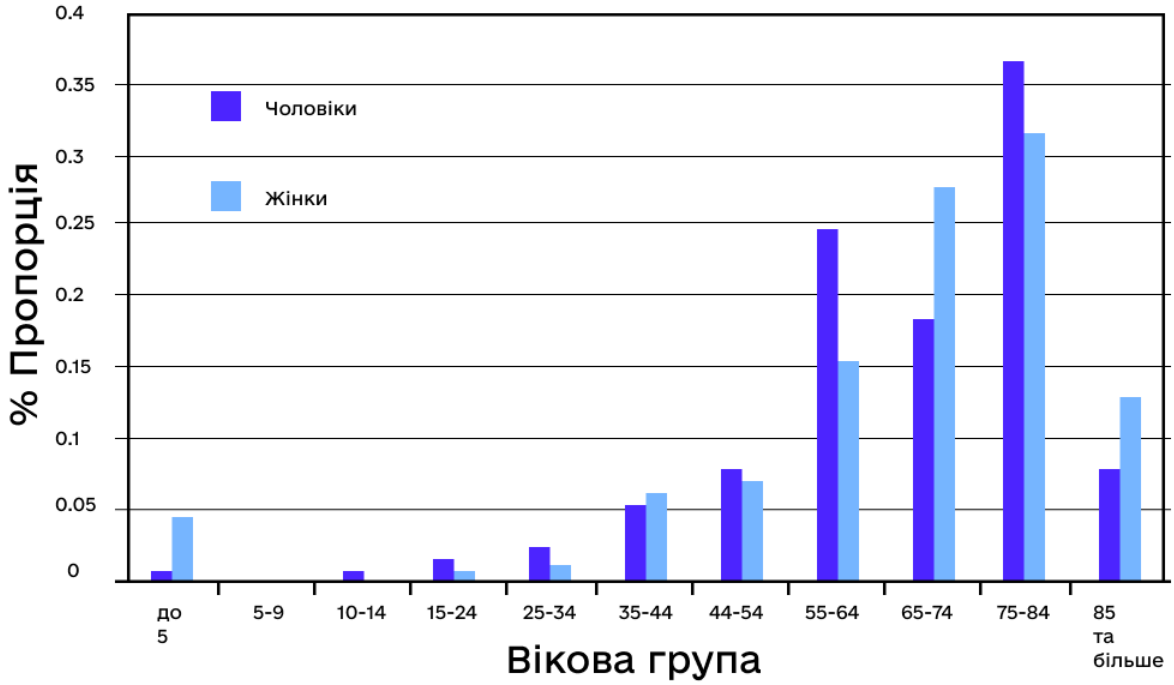

Рисунок 1.3 — Статево-віковий склад проспективно визначеної раптової серцево-судинної смерті

В загальній популяці, виникнення раптової серцевої смерті залежть від

віку, статі та наявності або відсутності в анамнезі серцево-судинних захворюваннь [1]. Згідно з матеріалами “Рекомендації Асоціації кардіологів України”, у чоловіків віком 60-69 років з попереднім анамнезом захворювання серця, рівень виникнення раптової серцевої смерті становить 8 на 1 тис населення / рік. У 25% пацієнтів, що померли раптово, раптова смерть є першим і єдиним проявом захворювання [1].

Приблизно в 10% випадків, за даними статистичних дослідженнь, причина раптової смерті невідома [5]. Говорячи про вікові категорії, на молодий вік припадає найбільший відсоток пацієїнтів, що помирають без діагностованого захворвання серця, в той час, як з епідемологічної перспективи, частота раптової серцевої смерті у молодих людей без визначених клінічних ознак серцевого захворювання є мізерною, де на цю категорію припадає відносно маленька кількість випадків, в порівняні з кількістю, яка спостерігається на загальній популяції [2]. Аналізуючи епідемологію, можна зробити висновок, що жертвами раптової серцевої смерті є велика кількість людей, які попередньо не мали діагностованих серцевих захворюваннь, що несе за собою цілком трагічні наслідки для суспільства та близьких [2].

1.3 Механізми розвитку аритмій

Однією з ключових причин раптової серцевої смерті є виникнення фатальних серцевих аритмій [6]. Аритмії характеризуються порушенням нормальної електричної діяльності серця, що в внаслідок призводить до незворотного припинення ефективної скоротливої функції [2]. Розглядаючи механізми виникнення аритмій в контексті цієї роботи, ми детальніше зосередимось на компонентах, таких як ішемія міокарда, ремоделювання серцевої тканини та електрофізіологічних аномаліях.

Ішемія міокарда виникає при недостатньому кровопостачанні

серцевого м'яза, призводячи до дефіциту кисню та накопичення метаболічних продуктів, що спотворює нормальне функціонування клітин [7]. Як наслідок, нестача кисню порушує баланс між деполяризацією та реполяризацією кардіоміоцитів (м’язових клітин серця, що є структурною одиницею міокарду). Це стимулює появу аномальних електричних імпульсів. Ці зміни можуть проявлятися як ранні або затримані післядеполяризації, що стають каталізаторами подальшої неконтрольованої електричної активності [7].

Супутнім етапом ішемії міокарда є ремоделювання серцевої тканини, що є відповіддю організму на ішемічні пошкодження [8]. Цей процес супроводжується формуванням рубцевої тканини, фіброзом та гіпертрофічними змінами, які призводять до виникнення окремих ділянок з неоднорідною провідністю. В таких ділянках створюються умови для утворення замкнених електричних кіл, які ініціюють самопідтримуюче поширення електричних імпульсів і створюють базу для розвитку фатальних аритмій [9].

Причиною таких електрофізіологічних аномалій є порушення роботи іонних каналів, зокрема каналів для натрію, калію та кальцію. Генетичні мутації або набуті дисфункції цих каналів спричиняють зміни у швидкості проведення імпульсів та тривалості потенціалу дії, що призводить до ненормального збудження. Це впливає на спроможність серця координувати свої скорочення, наслідком чого є фактично розвиток небезпечних аритмічних епізодів [10].

Електрофізіологічні розлади може призводити підвищена симпатична активація, яка спостерігається в умовах стресу та під впливом гормональних змін, оскільки надмірна секреція адреналіну підвищує збудливість кардіоміоцитів.

Підвищена симпатична активація, що часто спостерігається в умовах стресу або під впливом гормональних змін, також може поглиблювати електрофізіологічні розлади.

Таким чином, комплекс взаємодіючих факторів – ішемія,

ремоделювання тканини та електрофізіологічні аномалії реалізує досить сприятливі умови та середовище для виникнення фатальних шлуночкових аритмій, що, як було зазначено, є однією з ключових причин раптової серцевої смерті.

1.4 Взаємодія структурних та електрофізіологічних факторів

Часто, ризик розвитку фатальних аритмій формується в процесі інтегрованої взаємодії структурних змін міокарду та електрофізіологічних аномалій. Навіть за умов, в клінічному розумінні, нормального серця, незначні субклінічні морфологічні дефекти можуть взаємодіяти з порушенням у роботі іонних каналів, про яке ми згадували раніше. В наслідок, створюється елекрично нестабільний субстрат, що значно підвищує ризик раптової серцевої смерті.

При стандартних методах діагностики можуть залишатися непомічинеми структурні зміни, такі як дрібний фіброз, рубцювання та гіпертрофічні перетвореннння, які в свою чергу істотно впливають на розподіл електричної провідності, утворюючи зони з уповільненим проведенням імпульсу, або ж взагалі, з повним блокуванням, що відображається на аритмічних подіях.

1.5 Електрокардіографічні характеристики

Аналізуючи серцевий цикл здорвого організму, розглянемо послідовність подій. Основою серцевого циклу є електрична активність, яка координує та регулює скорочення серцевого м’язу (рис. 1.4).

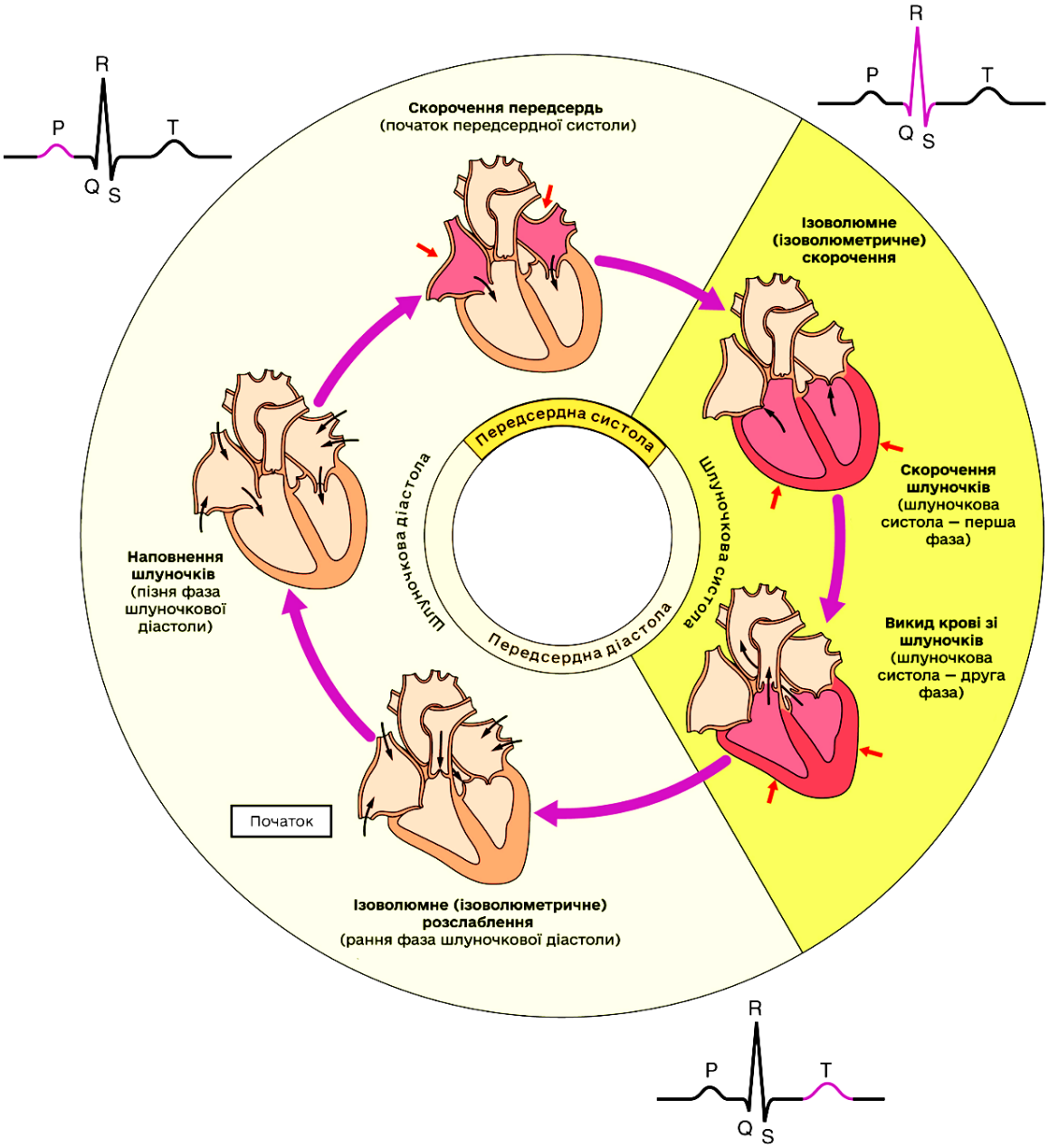

Рисунок 1.4 — Огляд серцевого циклу

Серцевий цикл починається з деполяризації - процесу, де мембранний потенціал клітини раптово змінюється від негативного до позитивного значення. Деполяризація ініціюється відкриттям натрієвих каналів, що дозволяє іонам натрію швидко потрапляти в клітину, в результаті чого, електричний імпульс миттєво поширюється по серцевій тканині, забезпечуючи синрхонне скорочення синоатріального вузла, передсердь та

шлуночків.

Після деполяризації, клітини повинні повернутися до свого початкового стану. Це повернення відбувається на етапі реполяризації – процесу, коли негативний мембранний потенціал клітин відновлюється за рахунок відкриття калієвих каналів та виходу іонів калію з клітини. У деяких клітинах, в цьому процесі беруть участь кальцієві канали, які визначають тривалість потенціалу дії. Таким чином, реполяризація є своєрідним підготовчим етапом до наступого циклу деполяризації та акту скорочення.

1.6 Електрокардіографічні предиктори раптової серцевої смерті

Розглядаючи електрокардіографічні показники, існує серія параметрів, які є прогностично цінними для визначення станів, що вказує на значний ризик раптової серцевої смерті. Сюди включають подовження інтервалу QT, дисперсію інтервалу QT, синдром Бругада та альтернції зубця Т.

1.6.1 Предиктор 1 (Подовження інтервалу QT)

Подовження інтервалу QT репрезентє порушення нормальної послідовності електричних подій у серці. На фізіологічному рівні, при подовженні QT, відновлення електричного потенціалу під час реполяризації затягується. Причиною цього визначають порушення роботи іонних каналів, відповідальних за рух натрію, калію та кальцію через клітинні мембрани. Як наслідок, час відновлення клітин міокарду збільшується та процес реполяризації набуває статусу неоднорідності на різних ділянках серцевого м’язу. Цей дисбаланс може призвести до виникнення ранніх післядеполяризацій (з англ «Early afterdepolarizations»), які, в свою чергу, призводять до аномальних серцевих ритмів, тахікардії та інших аритмій.

На електрокардіограмі, подовження інтервалу QT спостерігається як збільшення тривалості часу від початку комплексу QRS до завершення зубця Т, у порівнянні з нормальним станом (рис 1.5), де цей інтервал має встановлені межі.

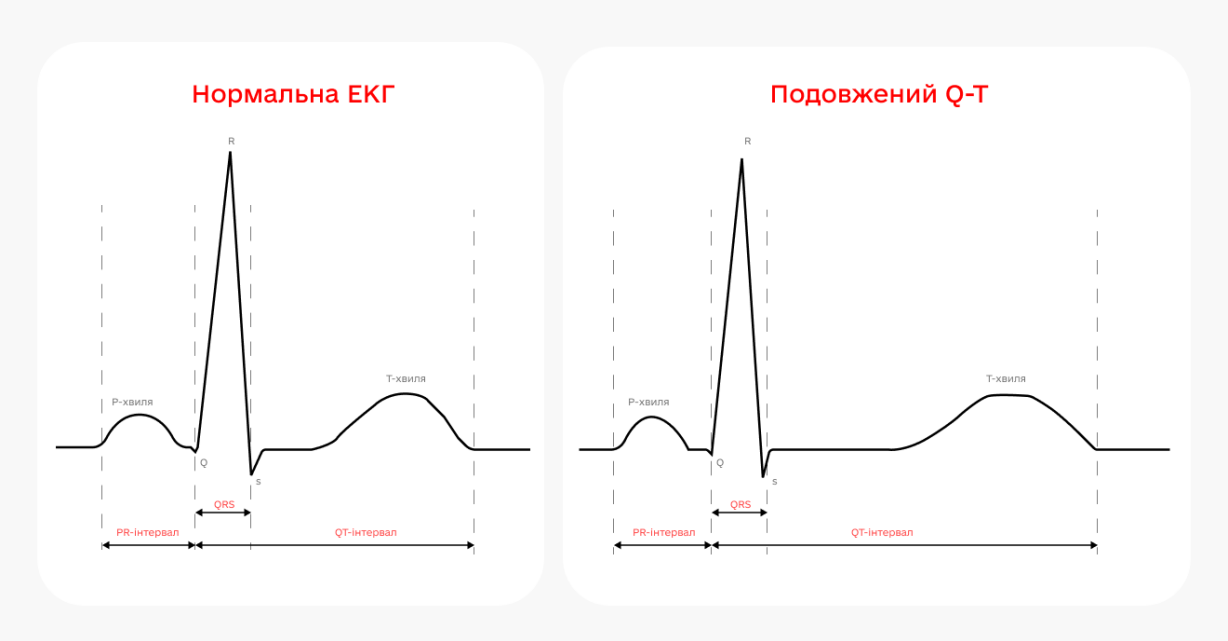

Рисунок 1.5 — Зображення нормальної ЕКГ зліва з позначеними нормальними інтервалами. Зубець P означає активність передсердь, QRS — активацію шлуночків, а інтервал QT — реполяризацію шлуночків

Розглядаючи діагностичну цінність данного предиктору, коли процес реполяризації порушений, електрична активність стає нестабільною, що спричиняє неконтрольоване поширення імпульсів і у разі стійкості даної аритмії, переходить у фібриляцію шлуночків. Ризик раптової серцевої смерті для пацієнтів при LQTS важко оцінити кількісно, оскільки існують певні генотипи, які в певних вікових групах мають вищий ризик РСС при данному синдромі. В чоловіків частота подій вища у молодому віці, жінки мають вищу захворюваність у дорослому віці.

1.6.2 Предиктор 2 (Дисперсія QT)

Дисперсія інтервалу QT відображає нерівномірність процесу реполяризації шлуночків у різних ділянках серцевого м’язу. Фізіологічно, тривалість відновлення електричного потенціалу залежить від локальних властивостей клітин та тканин: у здоровому серці всі ділянки міокарда реполяризуються приблизно синхронно, тому інтервали QT у різних

відведеннях ЕКГ практично однакові. Однак при захворюваннях, таких як ішемія, гіпертрофія або фіброз, окремі області серця можуть затримувати реполяризацію через порушення кровопостачання чи структурні зміни. Це призводить до того, що у деяких відведеннях QT-інтервал виявляється довшим, а в інших — коротшим, створюючи різницю між максимальним та мінімальним значенням.

На електрокардіограмі дисперсію QT побачити візуально складно, але її проявом є розбіжність у тривалості інтервалу від початку комплексу QRS до завершення зубця T серед різних відведень. У стандартній 12-канальній ЕКГ ця розбіжність може сягати десятків мілісекунд. Збільшення дисперсії означає, що окремі ділянки шлуночків із запізненням завершують реполяризацію, що створює нерівномірний електричний субстрат [11].

На рис 1.6 три криві демонструють різні шари міокарда:

- Субепікардіальні клітини — мають найшвидшу реполяризацію, тому їхній потенціал відновлюється раніше;
- Субендокардіальні клітини— реполяризуються трохи пізніше, їхня крива опускається повільніше;
- Клітини М-шару — демонструють найповільнішу реполяризацію, що призводить до найпізнішого відновлення їхнього потенціалу.

Вертикальні лінії позначають початок і кінець QT на звичайному ЕКГ-відведенні. Різниця між лівою та правою лініями і є дисперсією.

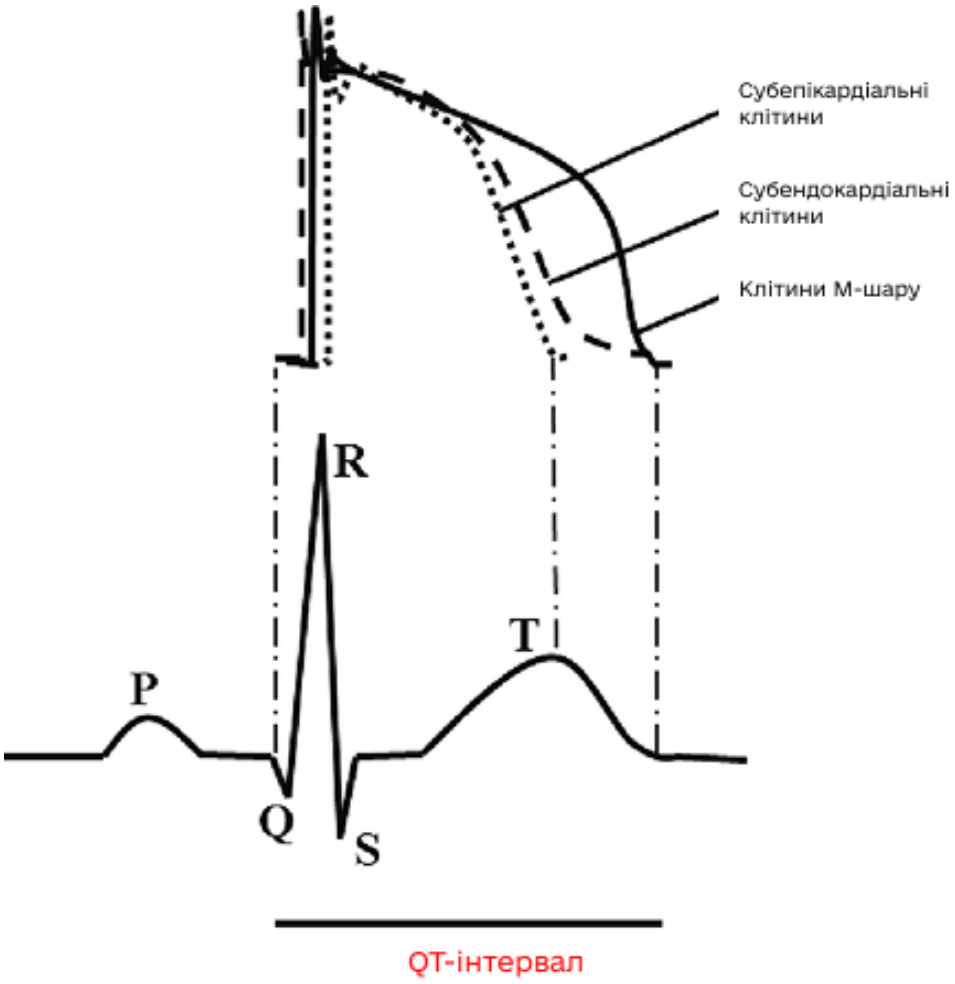

Рисунок 1.6 — Дисперсія QT: відмінність у часі завершення реполяризації субепікардіальних, субендокардіальних і М-клітин та її відображення на інтервалі QT [12]

Клінічно підвищена дисперсія QT асоціюється саме з підвищеним ризиком шлуночкових аритмій та відповідно з ризиком розвитку раптової серцевої смерті. Оскільки нерівномірність реполяризації створює зони з різною збудливістю, утворюються замкнені циркуляторні хвилі та ймовірність виникнення фатальних аритмій зростає. Дисперсія QT є незалежним предиктором подій, що дозволяє клініцистам більш точно оцінювати ризики та визначити пацієнтів, яким може бути рекомендована профілактика.

1.6.3 Предиктор 3 (Синдром Бругада)

Синдром Бругада є спадковим захворюванням іонних каналів серця

(каналопатія), яке не пов’язане зі структурними ураженнями міокарда безпосередньо, однак супроводжується високим ризиком швидкоплинних фатальних шлуночкових аритмій та раптової серцевої смерті. Його ключовою ознакою є специфічний вигляд сегмента ST у передніх прекордіальних відведеннях (V1–V3) на звичайній 12-канальній ЕКГ.

У загальному випадку (тип 1) на ЕКГ у відведенні V2 спостерігається підйом ST ≥ 2 мм із характерною формою так званої «опуклої купи» (coved), після якої зубець Т може бути плоским або негативним (рис. 1.7). Іноді зустрічається менш виражена «сідлоподібна» форма (saddleback), коли після невеликого підйому ST спостерігається короткий спад і другий невеликий підйом перед Т-хвилею [13].

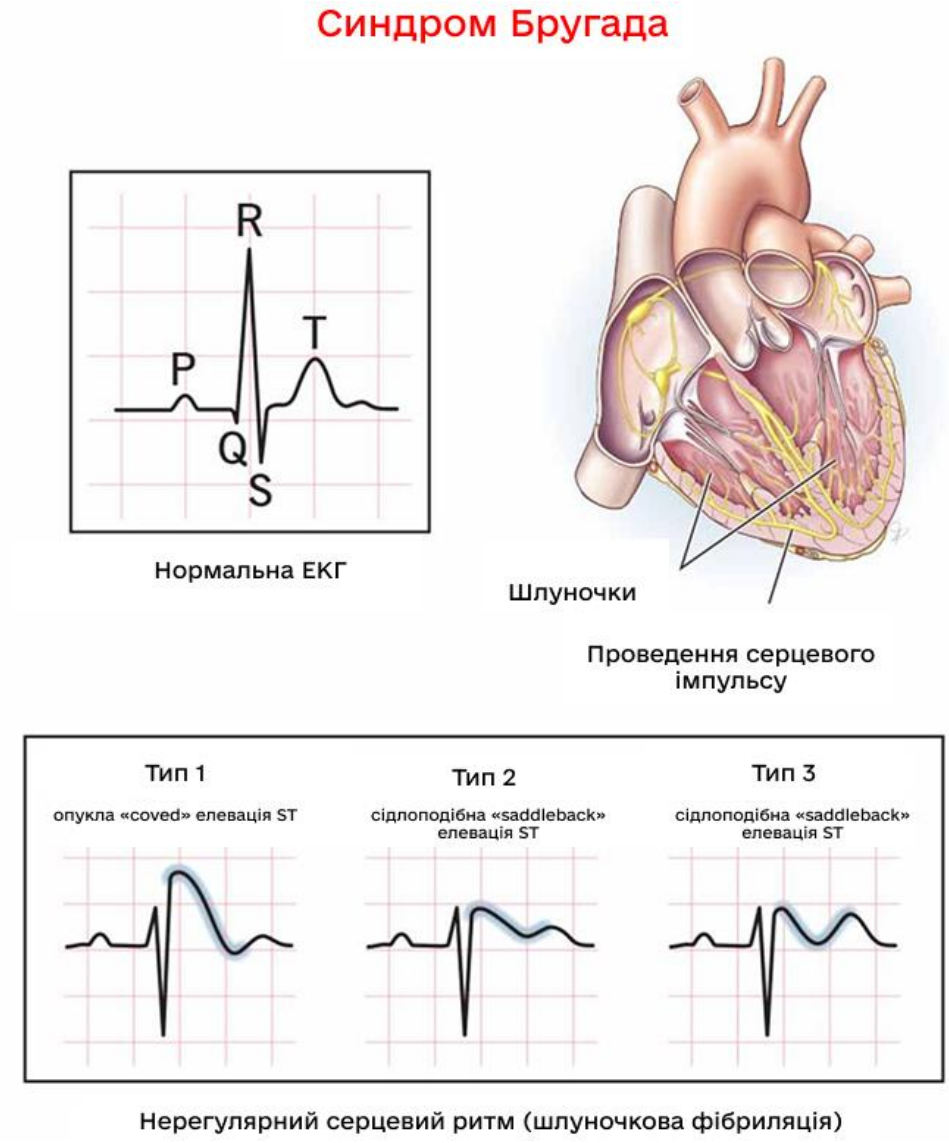

Рисунок 1.7 — Приклади “coved” та “saddleback” елевації ST в

прекордіальних відведеннях при синдромі Бругада

На практиці, лікарі звертають увагу саме на ці два варіанти:

- Coved-тип (тип 1): класичний опуклий «горбець» ST-сегмента із подальшим спуском у напрямку Т-хвилі.
- Saddleback-тип (тип 2): коротка послідовність підйом–спад–підйом ST перед Т-хвилею, що нагадує сідло.

При виявленні на поверхневій ЕКГ типового «coved» ST-зсуву у правих прекордіальних відведеннях навіть у клінічно асимптомного пацієнта слід запідозрити синдром Бругада та негайно провести подальше обстеження. Патофізіологічно, цей феномен відображає локальну втрату так званої «куполореполяризації» (action-potential dome) в епікардіальному шарі правого шлуночка. Такий градієнт іонних струмів може запустити короткі «re-entry» кола, що миттєво переходять у поліморфну шлуночкову тахікардію або фібриляцію — безпосередню причину раптової серцевої смерті [13].

При підтвердженні «coved» ST-зсуву призначають додаткову діагностику у вигляді тестів з натрієвими блокаторами, електрофізіологічне дослідження і генетичний скринінг родичів. Для пацієнтів із високим ризиком: синкопе, відповідний сімейний анамнез - основною методикою первинної профілактики раптової смерті є імплантація кардіовертера-дефібрилятора [13].

1.6.4 Предиктор 4 (Альтернація зубця Т)

Альтернація зубця Т являє собою явище, при якому форма та/або амплітуда зубця Т на електрокардіограмі змінюється від одного серцевого циклу до наступного (рис. 1.8) [14]. TWA поділяють на макроальтернацію, яка є візуально помітною на електрокардіографічних сигналах та мікроальтернацію, ідентифікація якої є неможливою людському оку та вимагає застосування високочутливих методів аналізу для детекції. Згідно з Консенсусними рекомендаціями Міжнародного товариства хольтерівської та неінвазивної електрокардіографії, грунтуючись на широкому масиві доказів і проспективних дослідженнь у >12000 пацієнтів, TWA є незалежним

предиктором шлуночкових аритмій та може застосовуватись для оцінки ризику раптової серцевої смерті [15]. Розглядають три патафізіологічні механізми, які викликають появу електричних альтернатів: альтернати, що стосються серцевого руху, альтернати провідності та альтернати реполяризації [16]. Альтернацію зубця Т відносять до останньої категорії.

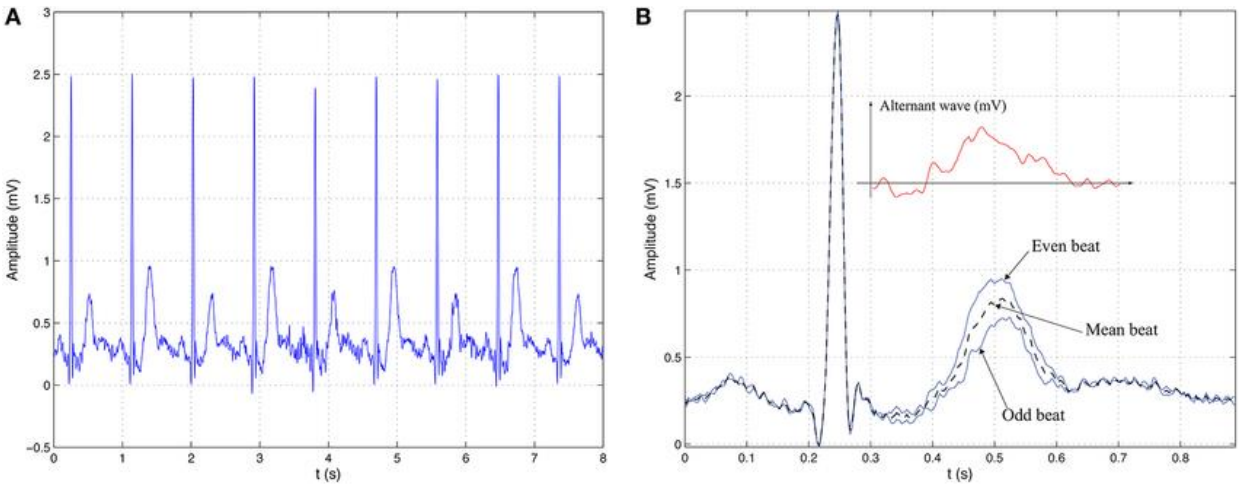

Рисунок 1.8 — Наочний приклад TWA. (A) ЕКГ-сигнал з періодичною зміною Т зубця в сегменті реполяризації з періодом в два удари. (B) Візуальна інтерпретація TWA як різниця між усередненим биттям події та усередненим непарним биттям

На електрокардіограмі, TWA виникає внаслідок чергування тривалості потенціалу дії на рівні серцевого міоцита. За нормального функціонування скороченнь серцевого м’язу, реполяризація забезпечує відновлення електричного потенціалу кардіоміоцитів після скорочення, таким чином, готуючи їх до наступного циклу деполяризації. У випадку з альтернацією зубця Т, замість сталого відновлення потенціалу дії, відбуваються циклічні коливання, які репрезентують дисфункцію регуляції внутрішньоклітинного обміну кальцієм і неоднорідність реакції клітин на внутрішньоклітинні сигнали [17]. Клітини міокарду відновлюють свій електричний потенціал із затримками та різницею в швидкості між окремими ділянками серцевої тканини, що призводить до зміни реполяризаційних характеристик ділянок з

порушенням кальцієвого обміну, в наслідок чого і виникають альтернати. Альтернацію зубця Т на електрокардіограмі спостерігається як послідовні зміни амплітуди або морфології зубця Т. За нормальних умов, зубець Т має стійку конфігурацію, яка відображає однорідну реполяризацію шлуночків, коли при наявності альтернації, ці характеристики порушуються. Такі порушення створюють субстрат для виникнення ранніх післядеполяризацій, що можть ініціювати шлуночкову тахікардію або фібриляцію. Оскільки альтернація передує іншим критичним станам, діагностичну цінність цього предиктору можна визначити у його здатності виявити електричну нестабільність на досить ранніх стадіях патолоічних змін серцевої тканини.

1.7 Порівняння альтернації зубця Т з іншими предикторами

Розглядаючи порівняльні характеристики предикторів раптової серцевої смерті, можна зауважити, що жоден із них не є ідеальним сам по собі. Кожен ЕКГ-маркер відображає різний аспект електричної нестабільності міокарда, тому в клінічній практиці для аналізу, зазвичай відбувається застосування комбінації декількох маркерів. В даному підрозділі проаналізуємо характеристики розглянутих предикторів, аби визначити предмет подальшого обговорення в цій роботі.

Синдром подовженого інтервалу QT відомий як один із найпоширеніших та найдоступніших предикторів. Він досить легко класифікується на будь-якій стандартній ЕКГ, однак коли змін у QT-інтервалі небагато, показник не спроможний забезпечити достатню частоту виявлення обстежуваних із підвищеним ризиком.

Наступним розглянутим показником була дисперсія QT, яка демонструє середню чутливість і вищу специфічність порівняно із синдромом подовженого QT: маркер аналізуєтсья з метою виявлення просторової гетерогенності реполяризації шлуночків. Для забезпечення надійності розрахунку та класифікації, прийнятним стандатом є застосування

запису усіх 12 каналів, корекціїї частоти серцебиття та програмного забезпечення для аналізу.

При розглянутому синдромі Бругада (тип 1) має низьку чутливість, оскільки зустрічається досить рідко. Також, виходячи з вищеописаних мною характеристик данного синдрому, його специфічність досить висока: її реєстрація одразу вказує на вроджену іонну каналопатію і вимагає негайних профілактичних заходів. При чому, в більшості випадків, додаткових тестів не потрібно — типовий малюнок ЕКГ може бути достатнім для діагнозу.

Синдром альтернації зубця Т комбінує в собі досить високу чутливість до ранніх порушень реполяризації. Його досить зручно виявляти в амбулаторних умовах: під час холтерівського моніторингу або виконання навантажувального тесту. Фізіологічно, альтернація зубця Т відображає електричну нестабільність, яка є первинним механізмом розвитку шлуночкових аритмій. Синдром може з’являтися за довго до виражених змін інших параметрів, що дозволяє збільшити часовий простір для інтервенції.

Прагнучи якомога більш ранньої та чутливої діагностики електричної нестабільності міокарда, в контексті цієї роботи, розглядатимемо альтернацію зубця Т, як доречний предиктор для комп’ютерної імплементації та подальшого автоматизованого аналізу в нашому програмному застосунку.

1.8 Методи моделювання альтернації зубця Т

Феномен альтернації зубця Т є відносно рідкісним електрофізіологічним явищем, що спостерігається лише у вузьких клінічних умовах. Це суттєво ускладнює доступ до відкритих або стандартизованих масивів ЕКГ-даних, у яких така морфологія представлена у вираженій формі. У зв’язку з цим, під час подальшого дослідження, ми розглядатимемо математичні алгоритми для відтворення властивостей чи результатів процесів реполяризації та деполяризації з метою отримання реперезентативних наборів даних для подальшого аналізу. На цьому етапі,

важливим буде поєднання двох компліментанних напрямів нашої роботи: створення контрольованих сигналів із заданими рівнями альтернації та можливості використання різноманіття екземплярів сигналів з відмінними один від одного параметрами для тестування широкого спектру різноманітних сценаріїв на імплементованих алгоритмах, розробка яких є другою невід'ємною складовою [18].

Моделювання уможливлює отримання нами репрезентативних наборів даних, де відомі істині значення рівня альтернації. Класичним підходом до відтворення властивостей реполяризації та деполяризації у дослідженнях електрофізіології є біофізичні моделі, що грунтуються на формалізмі Ходжкіна-Хакслі. Однією з перших таких моделей для шлуночкових міоцитів стала робота Біллер-Ройтера 1977 року, яка описує математичну модель мембранних потенціалів дії міокардіальних волокон шлуночків, складний процес реконструкції потенціау дії волокон міокарда шлуночків, де за допомогою відповдіних рівнянь, демонструвалося відтворення основних фаз потенціалу дії. Модель продемонструвала здатність відтворювати вплив співвідношення різних конфігурацій, зокрема на форму зубця Т [19]. У 1991 році, з'явилося дослідження з описом створення моделі потенціалу дії шлуночків серця, яке розширило спектр описуваних процесів Біллером-Ройтером, доповнивши модель врахуванням внутрішньоклітинного обміну кальцієм, динаміку насосів та .ін. [20].

Ці моделі набули статусу певного стандарту для детального вивчення серцевої електроізіології й дозволили імітувати просторово-часові збудження та реполяризації в тканинах. Велика кількість рівняннь та число параметрів значно ускладнює застосування відповідних моделей в реальному часі, підвищуючи обчислювальні вимоги та обмежуючи їхню реалізацію без доступу до високопродуктивних кластерів.

В якості альтернативного підходу, розглянемо інші моделі, імплементувавши які, ми матимемо можливість досягти компромісу між рівнем складності реалізації та фізіологічною достовірністю. Однією з таких

моделей є генеративна модель породження електрокардіограм [21]. У межах генерації ЕКГ, кожен серцевий цикл m описується як сума п'яти асиметричних гауссових імпульсів (P, Q, R, S, T), та шумових компонент.

Математично, сигнал

$$z_m(t) = \sum_{i \in \{P,Q,R,S,T\}} A_{i,m} exp\left(-\frac{(t-\mu_i)^2}{2b_i(t)^2}\right) + h_I^{(m)}(t) \qquad (1.1)$$

$$b_i(t) = \begin{cases} b_i^{(1)}, & t < \mu_i, \\ b_i^{(2)}, & t > \mu_i, \end{cases} \qquad (1.2)$$

де

- $A_{i,m}$ - амплітуда i-ї хвилі в m-му циклі,
- μ_i – часова позиція піку i,
- $b_i^{(1)}$, $b_i^{(2)}$– «ліва» та «права» ширини хвилі,
- $h_I^{(m)}(t)$ - внутрішній шум (високочастотні флуктації)

Кожна хвиля моделюється гауссовим імпульсом з двома різними дисперсіями ліворуч і праворуч від центру μ_i, що гарантує асиметрію форми. Відповідні ширини $b_i^{(1)}$, $b_i^{(2)}$ визначають, наскільки швидко імпульс зростає та спадає до і після піку. Сума таких імпульсів по п'яти складає PQRST-цикл [21].

З метою відтворення TWA, амплітуди Т-хвилі в кожнму циклі коригуються наступним чином:

$$A_{T,m} = A_T(1 + (-1)^m \delta_T) \qquad (1.3)$$

де δ_T – відносний рівень альтернації, а $(-1)^m$- чергує збільшену і зменшену аплітуду в непарних та парних циклах [21].

Внутрішній шум $h_I^{(m)}(t)$ моделюється як гаусівський білий шум з нульовим середнім і дисперсією σ_I^2:

$$h_I^{(m)}(t) \sim N(0, \sigma_I^2) \qquad (1.4)$$

Об'єднання обчислень дає загальний вираз для генерації синтетичного ЕКГ із чітко контрольованим рівнем альтернації та шумовою компонентою:

$$z_m(t) = \sum_{i \in \{P,Q,R,S,T\}} A_{i,m} exp\left(-\frac{(t-\mu_i)^2}{2b_i(t)^2}\right) + h_I^{(m)}(t) \qquad (1.5)$$

Така сума гарантує відтворення всіх клінічно значущих форм ЕКГ, забезпечує можливість аналізувати ефективність алгоритмів на широкому діапазоні змінених параметрів і дає змогу формувати різні сценарії для валідації чутливості методів ідентифікації альтернації зубця Т [21].

Висновок до розділу 1

У ході літературного аналізу уточнено сучасні дефініції раптової серцевої смерті та показано, що ключовою ознакою визнано нетравматичну, несподівану смерть кардіального генезу, яка настає протягом однієї години від початку симптомів. Епідеміологічні дані демонструють домінування ішемічної хвороби серця (≈ 80 % випадків) і виокремлюють чотири групи чинників ризику, серед яких особливу роль відіграють електричні канало- і реполяризаційні аномалії. Розглянуто фізіологію нормального серцевого циклу та показано, що дисбаланс деполяризації-реполяризації лежить в основі фатальних аритмій.

Систематизовано електрокардіографічні маркери, здатні сигналізувати

про підвищений ризик РСС: подовження і дисперсія QT-інтервалу, синдром Бругада та альтернація зубця Т. Порівняльний огляд засвідчив, що подовжений чи дисперсований інтервал і Бругада вирізняються або надто низькою чутливістю, або рідкістю проявів, тоді як альтернація Т-хвилі поєднує високу чутливість до ранньої електричної нестабільності з можливістю реєстрації під час холтерівського моніторингу чи навантажувальних тестів. Саме тому Т-альтернація обрана основним об’єктом подальшого комп’ютерного аналізу.

Оскільки клінічні записи з вираженою Т-альтернацією практично відсутні у відкритому доступі, обґрунтовано необхідність генерації синтетичних ЕКГ-сигналів. Оцінено два класи моделей: біофізичні (формалізм Годжкіна–Хакслі) і спрощені гаусівські генеративні схеми; останні забезпечують прийнятний компроміс між фізіологічною достовірністю та обчислювальною ефективністю, дозволяючи варіювати рівень шуму й амплітуду альтернації.

Розділ створює наукову й методологічну основу для розробки програмного інструменту, що синтезує контрольовані послідовності ЕКГ з заданим рівнем альтернації зубця Т та реалізує алгоритм автоматичного виявлення цього предиктора, що розглядатиметься в наступних розділах.

РОЗДІЛ 2
ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАСТОСУНКУ

В розділі розглядатиметься практичний інструментарій, на якому ґрунтується реалізація програмного застосунку: від загальної архітектури та вибору мови програмування до форматів подання даних, бібліотек візуалізації та підходів до машинного аналізу сигналів. Буде описана схематична будова системи й принцип розподілу обов’язків між її логічними шарами; узагальнюватиметься склад використовуваного технологічного стеку — мова програмування, середовище виконання, засоби роботи з графікою та відповідним графічним інтерфейом, базою даних і мережевими компонентами. Розглядатиметься модель обчислювального модулю, що забезпечуватиме оцінку показників альтернації. Визначатиметься процес складання й розгортання застосунку, демонструючи, як перелічені засоби інтегруються у цілісний, придатний до експлуатації програмний продукт.

2.1 Високорівнева архітектура

У рамках реалізації програмного комплексу, будемо застосовувати досить класичний трирівневий архітектурний стиль, який здійснює досить чітке та інтуїтивно зрозуміле розмежування функціональних обов’язків між програмними компонентами та спрощує подальшу підтримку й розширення системи.

На рис 2.1 наведено узагальнену архітектуру, що складається з трьох рівнів: рівень представлення, бізнес-логіки та рівень доступу до даних [22]. На рівні представлення буде реалізовуватись логіка взаємодії застосунку з користувачем. Реалізація графічного інтерфейсу та програмних компонентів здійснюватиметься за використанням фреймворку JavaFX та відповідних FXML-шаблонів сторінок. В даному контексті, існуватиме деякий клас,

AppLauncher що здійснюватиме ініціалізацію середовища та слугуватиме точкою входу в програму. У кожного шаблону сторінки існувтаиме власний клас-контроллер, в якому буде реалізована логіка обробки подій, які спричиняє користувач власною ваємодією з програмним інтерфейсом, здійснюватиметься передача параметрів генерації та аналізу до рівня бізнес-логіки та відображатимуться результати на графіках і візуальних компонентах [23].

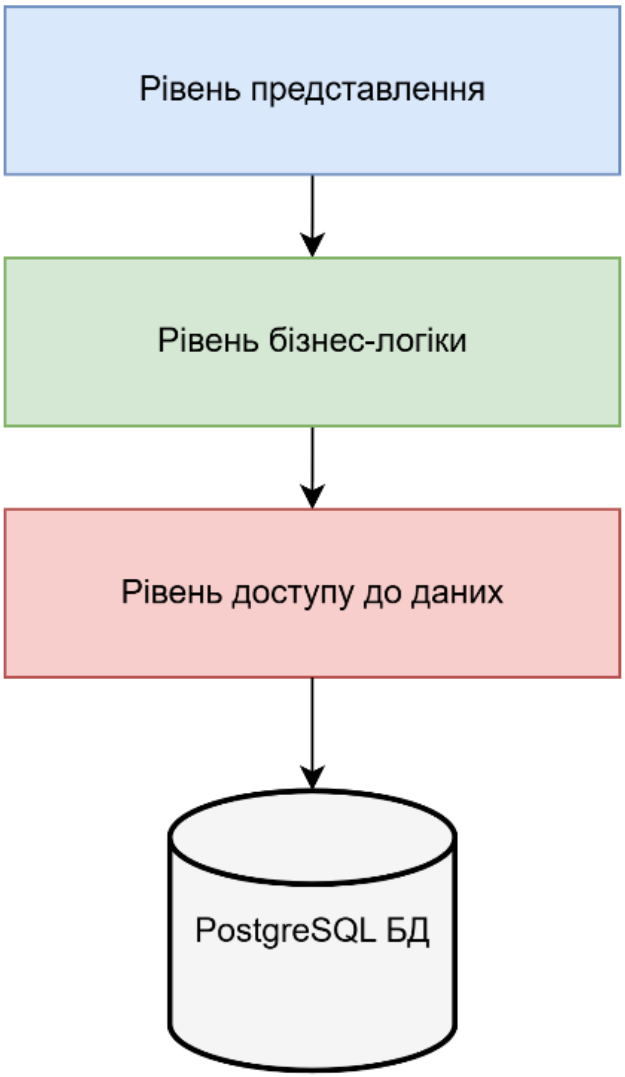

Рисунок 2.1 — Загальна архітектурна схема системи

Рівень бізнес-логіки інкапсулюватиме основну обчислювальну й

алгоритмічну частину системи [24]. На цьому рівні передбачені окремі класи, які генеруватимуть масиви PQRST-хвиль відповідно до налаштувань із контролера, класи для обчислення рівня альтернації зубця T та супроводжуючих метрик для глибшого аналізу сигналу. Також, на даному рівні буде впроваджена сервісна частина, яка координуватиме виклики генерації, аналізу та збереження послідовностей, формуючи більш високорівневі API для рівня представлення.

Рівень доступу до даних реалізовуватиме взаємодію застосунку з базою даних PostgreSQL використовуючи ORM-рішення (ORM — об'єктно-реляційне відображення) у вигляді JPA/Hibernate (рис. 2.2) [25].

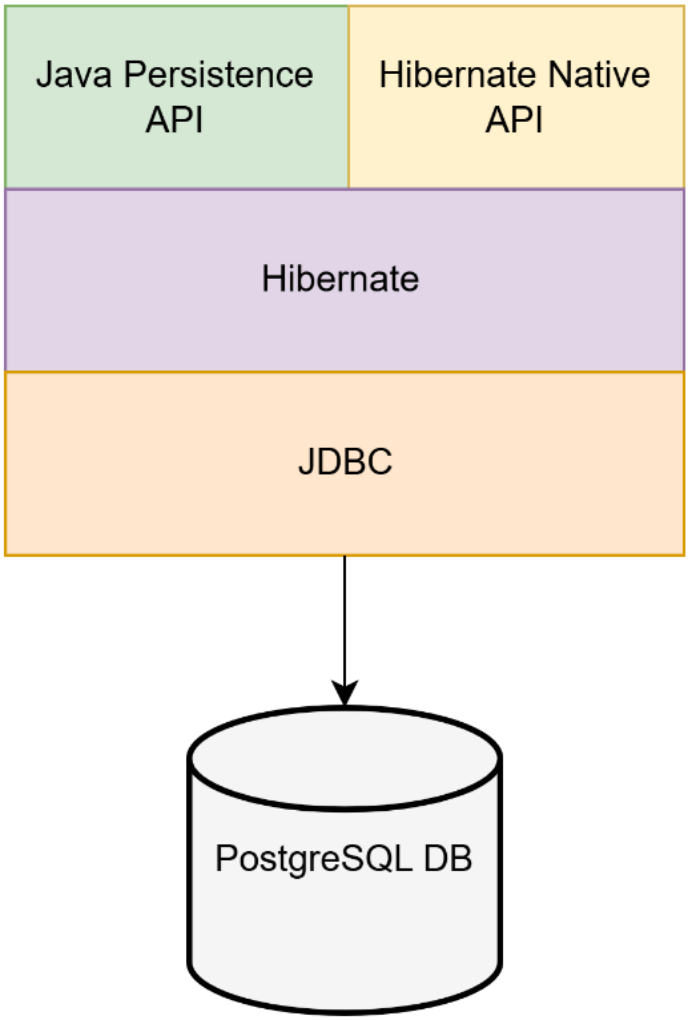

Рисунок 2.2 — Архітектура технологій рівня доступу до даних з використанням JPA/Hibernate

На рівні доступу до даних описуватиметься сутність EcgSequence, що міститиме поля для збереження сигналів у вигляді масиву байтів, а також відповідних кожній сутності метаданих: назва послідовності, рівень заданого шуму, рівень заданої альтернації, кількість точок у послідовності, кількість циклів. Для даної сутності буде реалізований клас DAO (Data Access Object — об'єкт доступу до даних), який описуватиме загальні, для роботи з базою даних, операції створення, читання, оновлення, видалення. В окремному пакеті на цьому рівні будуть реалізовані утилітарні класи з інфраструктурною функцією для налаштування пулу з’єднаннь JDBC (Java Database Connectivity — з’єднання з базою даних у середовищі Java) та параметри з’єднання з СУБД [26].

Міжшарова комунікація реалізовуватиметься через чітко визначені інтерфейси: контролери рівня відображення викликатимуть методи сервісів рівня бізнес-логіки, які у свою чергу використовують клас DAO для читання й запису сутностей. Архітектура застосунку дозволить нам тестувати кожний із рівнів незалежно, змінюючи DAO-класи на замінники реальних об’єктів, які імітуватимуть їхню поведінку за заданими сценаріями. В майбутньому, ми зможемо розширювати систему, інтегруючи нові модулі без втручання в інші компоненти [27].

2.2 Технологічний стек

Для реалізації програмного застосунку обрано мову програмування Java, яка є, свого роду, галузевим стандартом для крос-платформених застосунків. Вихідні файли .java компілюються засобом javac у байт-код формату .class. Цей проміжний об’єкт є архітектурно нейтральним, що дає нам змогу виконувати один і той самий артефакт без перекомпіляції на будь-якій платформі з сумісною JVM [28].

Під час виконання, віртуальна машина застосовує двоступеневий трансльований режим. Спершу байт-код інтерпретується, забезпечуючи

мінімальний час для старту. Далі підсистема JIT Compiler перетворює часто викликані ділянки коду на оптимізований машинний код цільового процесора, з використанням різноманітних оптимізацій [29]. Продуктивність такого рішення наближається до нативних C/C++ збірок. Відповідний показник швидкості є суттєвим для нас, оскільки забезпечує можливість інтерактивного рендерингу синтетичних ЕКГ та швидкого обчислення рівнів альтернації зубця Т. Також, Java має статус мови з довгостроковою підтримкою, який передбачає наявність цієї підтримки впродовж щонайменше восьми років [30].

Візуальний шар застосунк побудовано на тріаді JavaFX / FXML / CSS. JavaFX відповідає за формування графаічної сцени; FXML виконує роль декларативного опису компонентів, стилізацію та адаптивні відступи, винесені у зовнішні CSS таблиці, що в процесі реалізації дозволятиме нам підтримувати чистий модульний код і швидко модифікувати зовнішній вигляд без перекомпіляції [31]. Кожен екран оформлено однаковим файловим пакетом (рис. 2.3):

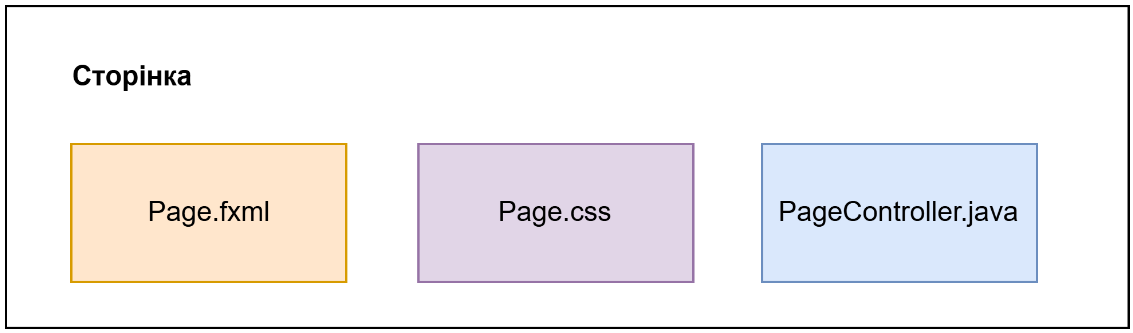

Рисунок 2.3 — Структура файлового пакету кожної сторінки

- **Page.fxml** — декларативна розмітка вузлів JavaFX;
- **PageController.java** — клас-контролер із логікою обробки подій;
- **Page.css** — локальні стилі, що перевизначають базову тему;

Для зберігання та вибірки даних, застосунок використовуватиме

ієрархію «JPA → Hibernate → JDBC → PostgreSQL» (рис. 2.4). На найвищому рівні Jakarta Persistence API виконує роль декларативного посередника, де класи з анотацією @Entity описують сутності, а поля цих класів автоматично інтерпретуються як стовпці таблиць. Далі, ці метадані обробляє Hibernate, що виступає конкретною реалізацією стандарту JPA, й у момент виконання, перетворює кожну операцію з об’єктом на оптимізовані інструкції SQL [32]. Для передачі згенерованих запитів до серверу бази даних, Hibernate використовує JDBC як більш низькорівневу технологію взаємодії застосунку з базою, через яку реалізується процес відкриття з’єднання, створюються Prepared-Statement-и, передаються параметри та фіксуються транзакції. У якості саме бази даних імплементована PostgreSQL, реляційна система, що забезпечує транзакційну цілісність, та підтримку великих бінарних об’єктів (в нашому разі, це масивів сигналів) [33].

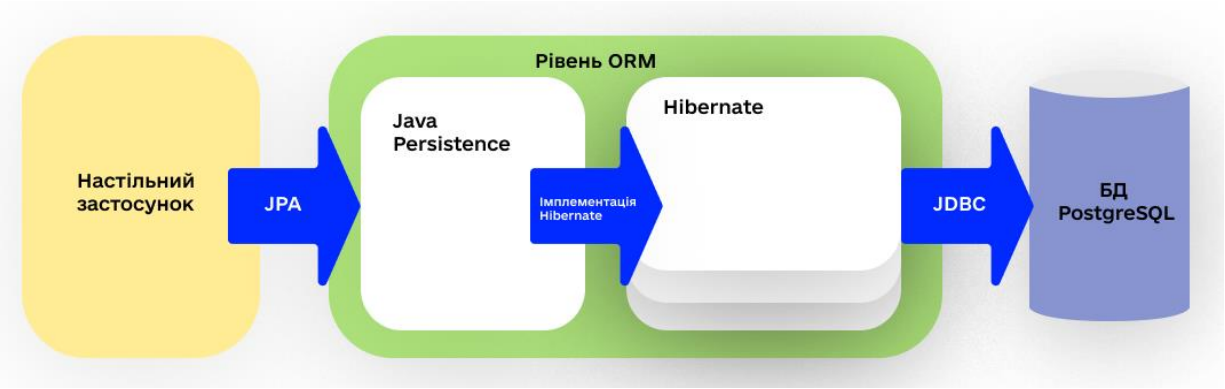

Рисунок 2.4 — Схема взаємодії JPA–Hibernate–JDBC

2.3 Компонентна діаграма та взаємодія

У цьому підрозділі розглядатимемо високорівневу компонентну діаграму програмного застосунку. Кожен із рівнів програми реалізовуватиме та виконуватиме свої чітко-окреслені функції та взаємодіятиме з іншими рівнями через задекларовані інтерфейси та служби [34].

На рис 2.5 – починаючи з найвищого рівня, розглянемо шар представлення, що відповідає за користувацьку взамодію.

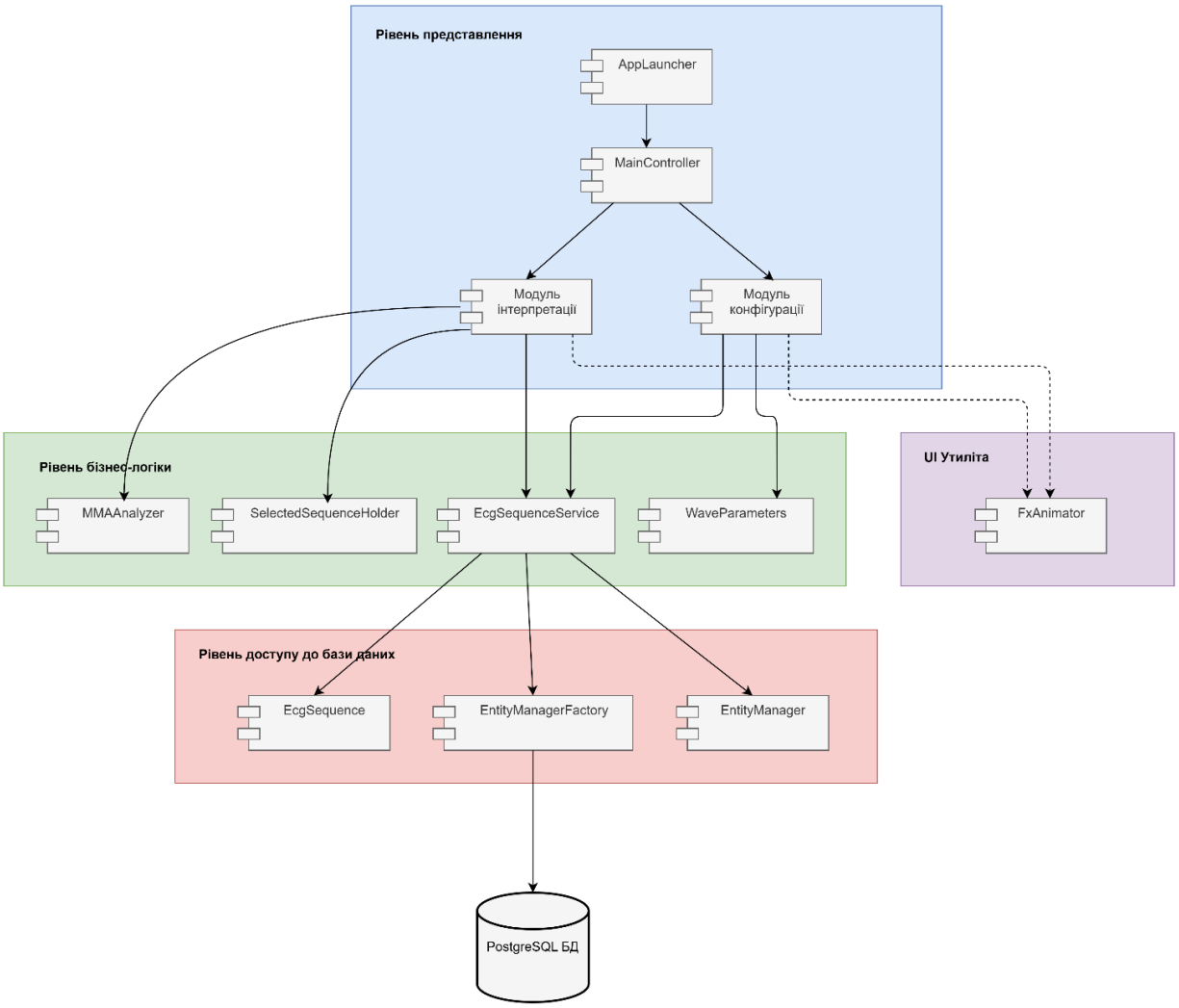

Рисунок 2.5 — Компонентна діаграма, що ілюструє взаємодію між Presentation, Business Logic та Data Persistence

Точка входу — клас AppLauncher, визначений стандартом графічного середовища JavaFX, ініціалізує JavaFX та завантажує головний контролер MainController. Головний контроллер керуватиме навігацією між двома підсистемами програмного застосунку: конфігураційний модуль, що відповідає за налаштування еталонного ЕКГ циклу та генерацію ЕКГ–послідовностей та модуль інтерпретації, в якому здійснюватиметься управління й аналіз збережених сигналів. Обидва модулі будуть

реалізовуватись, застосовуючи подібний патерн, де один контроллер матиме кілька дочірніх підконтроллерів, кожен з яких синхронізуватиме свою частину інтерфейсу з відповідними об’єктами рівня бізнес-логіки [35].

Наступний шар, рівень бізнес-логіки, здійснює інкапсуляцію обчислювальної та сервісної частини системи. Ключовим компонентом шару є EcgSequenceService – клас, що реалізовуватиме високорівневі операції CRUD (збереження, отримання списку, видалення та оновлення) для сутності EcgSequence [36]. Через цей сервіс, Presentation Layer викликатиме логіку збереження згенерованих сигналів та діставатиме їх для відображення відповідно. Паралельно з сервісом буде імплементований клас MMAAnalyzer, що реалізовуватиме алгоритм обчислення рівня альтернації зубця T та супроводжуючих метрик, і слугуватиме аналітичним модулем для нашого застосунку. Для передачі вибраних користувачем ідентифікаторів послідовності між рівнями, застосовується Singleton-компонент SelectedSequenceHolder, який фіксуватиме об’єкт послідовності електрокардіографічних сигналів, з якою в поточний момент взаємодіятиме користувач. З метою інкапсуляції стандартних параметрів еталонного ЕКГ циклу, реалізовуватиметься об’єкт WaveParameters [37].

Третій шар, рівень доступу до бази даних, відповідатиме за безпосередню взаємодію з реляційною СУБД PostgreSQL за допомогою ORM фреймворків та їхньої реалізації – JPA/Hibernate [38]. Головною сутністю цього шару буде клас EcgSequence з полями id, name, cycles, tWaveAltLevel, noiseLevel та signalData (масив байтів, що зберігає значення згенерованого сигналу). Через клас EntityManagerFactory і сам EntityManager, сервіс EcgSequenceService відкриватиме транзакції, виконуючи операції збереження, оновлення та взяття та видалення, ізолюючи при цьому бізнес–логіку від низькорівневих деталей JDBC-викликів [39].

Окремим шаром є UI утиліта, що містить в собі компонент FxAnimator. Він не входить до жодного з інших шарів безпосередньо, але його методи викликатимуться контролерами рівня відображення для забезпечення

анімації графічних компонентів та покращення користувацького досвіду [40].

Описуючи взаємодію між шарами, визначимо узагальнені правила цієї взаємодії: контролери рівня відображення викликатимуть сервіси рівня бізнес-логіки, які, в свою чергу, звертаються до рівня з'єднання з БД через DAO–подібні виклики JPA. UI утиліта під'єднується до рівня відображення за допомогою простого імпорту та викликів анімаційних функцій в методах ініціалізації сторінки.

На даному етапі аналізу та формулювання вимог, описана вище архітектура та технологічні засоби створюють умови для забезпечення чіткої сегментації відповідальності дії того чи іншого програмного компонету, що дозволить нам безперешкодно здійснювати модульне тестування та масштабувати застосунок без нагальної необхідності втручання в існуючий програмний код інших рівнів.

2.4 Процес збірки та розгортання

Розробляючи настільні застосунки, важливим етапом є забезпечення плавного переходу від вихідного коду до стабільного виконуваного продукту, оптимізація інсталяційного процесу та запуск програми, зробивши його максимально простим для користувача. В нашому застосунку, ця задача вирішуватиметься двоетапною моделлю: автоматизована збірка через Maven (система автоматизованої збірки) та виконання застосунку в середовищі Java Runtime з локальною базою даних PostgreSQL [41].

Перший етап, процес збирання програмного застосунку, починатиметься в каталозі проекту, де розміщені всі Java-класи, FXML-шаблони, CSS-стилі й файл pom.xml, який містить опис залежностей (JavaFX, Hibernate/JPA, драйвер PostgreSQL та інші бібліотеки) а також конфігурацію плагінів для створення «fat JAR» (рис. 2.6).

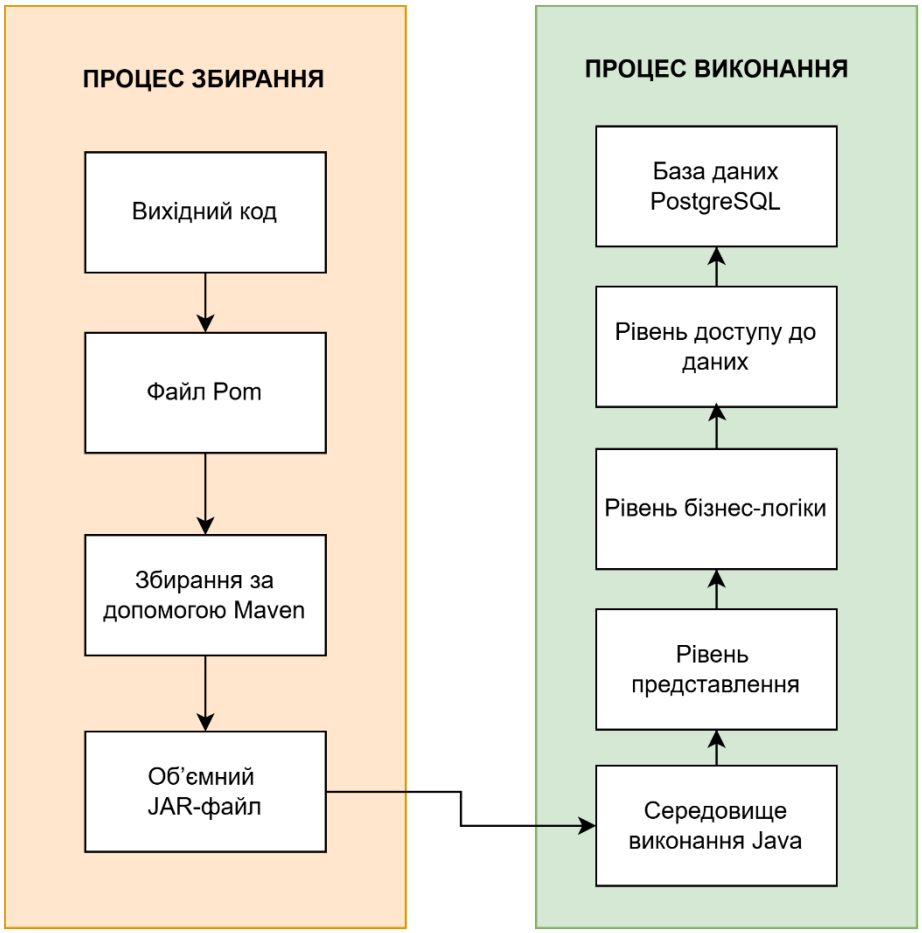

Рисунок 2.6 — Діаграма розгортання

Команда mvn clean package в терміналі автоматично завантажуватиме потрібні артефакти, здійснюватиме компіляцію програмного коду, оброблятиме ресурси та пакуватиме все в один єдиний виконуваний архів .jar, який міститиме сам додаток та всі сторонні бібліотеки, які в ньому застосовуються. Кінцевий артефакт даватиме нам гарантію того, що за наявності Java, він запусткатиметься без додаткових налаштувань практично на будь якому персональному комп'ютері [42].

Другим етапом є процес виконання, який ініціються шляхом написання відповідної команди в терміналі: java -jar App.jar, яка працюватиме на будь-якій системі з встановленою Java [42]. Після запуску команди, головний клас завантажуватиме JavaFX-сцену, зв'язуватиме контролери з FXML-шаблонами та демонструватиме користувачу наш інтерфейс, з яким потім

відбуваєтья користувацька взаємодія. Оскільки клієнт і сервер СУБД перебуватимуть на одному вузлі (localhost:5432 за замовчуванням у PostgreSQL) - усуваються будь які мережеві затримки й складність налаштувань мережевих з'єднань [43].

На етапі тестування та демонтрації виконання застосунку, ми зможемо виконати автоматизовану збірку проєкту в єдиний JAR-файл із максимально простою і прозорою схемою розгортання, де для нас, як розробників, або кінцевого користувача, головною вимогою буде наявність встановлених Java і PostgreSQL, перенесення JAR файлу у будь-яку директорію та його запуск відповідно [44].

2.5 Модель аналізу сигналу

В реалізації обчислювального сервісу для обробки ЕКГ-сигналів, використовуватимется алгоритм модифікованого ковзнього середнього (Modified Moving Average (MMA)) [25]. У наступних підпунктах поетапно розглянемо реалізацію цього алгоритму в нашому програмному застосунку.

Перед обчисленням рівня альтернації, сигнал проходить процедуру препроцесингу з метою отимання обчислювальним модулем даних про структуру сигналу, застосовуючи ці дані для інтерпритації вхідного сигналу та відповідного ухвалення рішення про застосування алгоритму (рис. 2.7). Спочатку ми ділимо вхідний ЕКГ-сигнал на послідовні цикли серцевих скорочень, що дозволить нам аналізувати кожен серцевий скорочувальний цикл окремо. У кожному такому фрагменті ми шукаємо найвиразнішу точку комплексу QRS, яку називаємо R-піком. Після цього, для кожного R-піку формуємо часовий інтервал, у якому розташована Т-хвиля. В цьому проміжку, алгоритм знаходить локальний екстремум сигналу (максимум або мінімум). Якщо локальних екстремумів у виділеному вікні немає, вибирається точка з найбільшою абсолютною амплітудою. Після того як усі Т-піки визначені, формуються впорядкований масив значень їх амплітуд (в

порядку слідування циклів). Щоб перевірити, чи справді відбувається чергування, характерне для явища альтернації, ми будуємо послідовність різниць між сусідніми амплітудами. Кожна така різниця показує динаміку зміни Т-амплітуди від одного циклу до наступного — чи зросла вона, чи зменшилася.

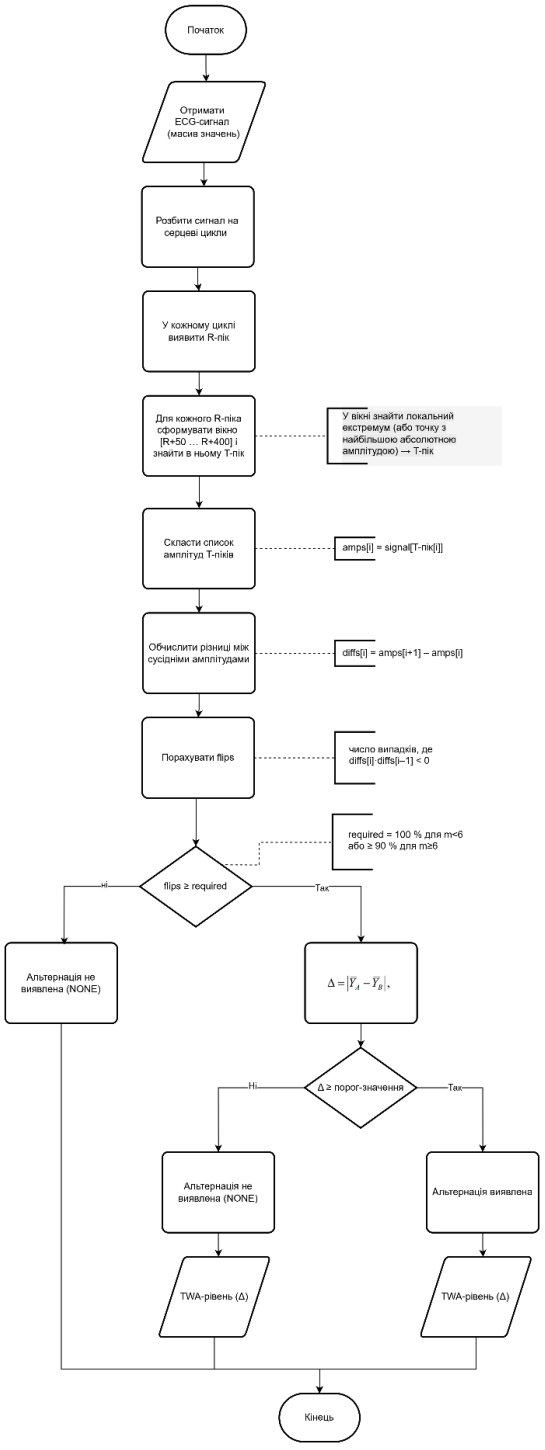

Рисунок 2.7 — Блок-схема обробки сигналу для виявлення рівня

альтернації

Наступним етапом є оцінка регулярності чергування. Ми підраховуємо, скільки разів в цій послідовності відбулося справжнє змінення знака різниці (зростання → спадання, або спадання → зростання). Якщо таких «фліпів» (в програмному коді, відповідний масив найменовано (fliping)) недостатньо, що свідчить про випадкові коливання, алгоритм одразу фіксує відсутність альтернації. Лише коли чергування є регулярним, алгоритм переходить до блоку розраунку рівня альтернації.

Визначається середня амплітуда парних та непарних Т-хвиль, після чого, рівень альтернації обчислюється як абсолютна різниця відповідних амплітуд, що відповідає ключовій ідеї ММА — розділення знайдених Т-хвиль на дві послідовності за індексом (парні й непарні) та розрахунок їхніх середніх амплітуд [46]:

$$\overline{T_{even}} = \frac{1}{N_{even}} \sum_{i\ even} s[t_i], \quad \overline{T_{odd}} = \frac{1}{N_{odd}} \sum_{i \in odd} s \qquad (2.1)$$

Після отримання T_k індексів, окремий програмний метод поділяє їх на парні та непарні сегементи. Для кожної підгрупи рахується середня амплітуда, після чого береться модуль різниці [45]:

$$\Delta = |\overline{T_{even}} - \overline{T_{odd}}| \qquad (2.2)$$

Результат обчислення і буде вважатись рівнем альтернації: якщо $\Delta < 0.02$ мВ — альтернація вважатиметься відсутньою, в іншому випадку, сигнал вважатиметься як такий, що містить альтернацію [45].

Висновок до розділу 2

У цьому розділі окреслено сукупність технічних рішень, що лежать в основі розробленого застосунку. Показано, як класична трирівнева архітектура, сучасна версія Java та перевірені бібліотеки утворюють цілісне середовище, здатне імітувати електрокардіографічні сигнали, зберігати їх у реляційній базі та виконувати автоматичний аналіз. Продемонстровано, що обраний стек забезпечує портативність, достатню обчислювальну швидкодію й прозорий процес розгортання, а застосування алгоритму MMA ілюструє можливість інтеграції доменно-специфічних методик без ускладнення загальної структури. Сукупність описаних засобів формує надійну платформу, придатну для подальшого масштабування й наукових експериментів.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ТА МЕТОДИКА РОБОТИ ПРОГРАМНОГО ЗАСТОСУНКУ

В третьому розділі розглядатиметься практична частина розробки програмного застосунку: від опису логіки переходів між екранами до демонстрації того, як алгоритм модифікованого ковзнього середнього (Modified Moving Average) на реальних викликах взаємодіє з базою даних і повертає користувачеві діагностичний результат. Оглядається ієрархія вікон, послідовно розглядається роль кожного шару та реалізується сценарій наскрізного тестування «генерування → збереження → аналіз»: для різних конфігурацій послідовностей.

3.1 Загальна ієрархія вікон та навігаційна логіка

У процесі розробки одним із ключових завдань була побудова гнучкої та роширюваної структури користувацького інтерфейсу. Для цього було обрано підхід із чітким розділенням розмітки, стилів і контролерів: кожен екран або діалог має власний файл FXML, який описує ієрархію JavaFX-вузлів, та файл CSS, який задає стилістичні правила для цього екрану.

Центральним елементом інтерфейсу є головне вікно Main.fxml, у якому розміщено дві основні вкладки – «Configuration» і «Interpretation» – та панель для динамічного завантаження вмісту (рис. 3.1).

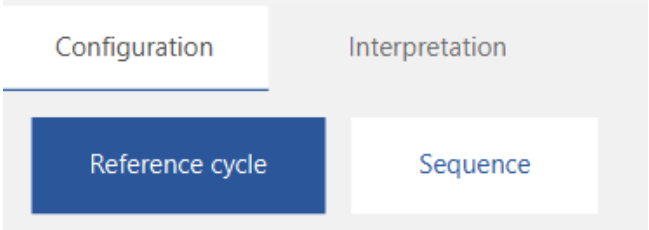

Рисунок 3.1 – Структура головного вікна (Main.fxml) та панель вкладок

Розмітка Main.fxml підключає загальний стиль style.css, та містить в собі елементи керування перемиканням розділів. По натисканню кнопки «Configuration», у контейнер підвантажується файл Configuration.fxml, а при переході на «Interpretation» – файл Interpretation.fxml. Кожен із цих FXML-файлів, у свою чергу, містить власні вкладені області для ще тоншої навігації: Configuration.fxml забезпечує внутрішнє перемикання між ReferenceCycle.fxml і Sequence.fxml, а Interpretation.fxml містить три таби для CurrentSequence.fxml, ManageSequences.fxml та DetailedAnalysis.fxml. Обраний ієрархічний підхід до організації розмітки гарантуватиме, що кожен екран відповідає за свій набір завдань і може бути змінений або протестований незалежно від інших. На рис. 3.2 продемонстровано функціональну ієрархію головних вікон інтерфейсу:

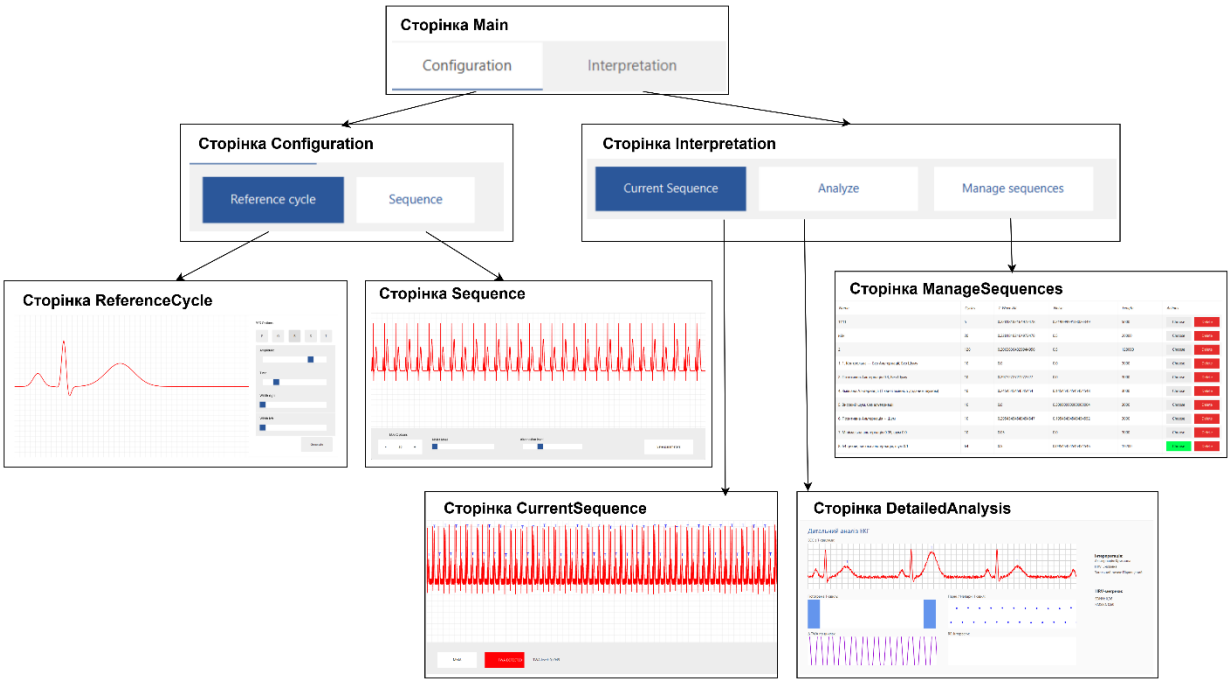

Рисунок 3.2 — Ієрархія головних вікон інтерфейсу, що реалізована згідно з проєктуванням в розділі 2

У FXML-файлах використано атрибути fx:controller і fx:id, що дозволяють впроваджувати елементи розмітки безпосередньо у поля контролерів [43]. Наприклад, у ReferenceCycleController.java прив’язуються

слайдери та Canvas, у SequenceController.java – елементи для вибору кількості циклів і рівнів шуму чи альтернації, а в DetailedAnalysisController.java – низка Canvas-компонентів для побудови гістограм та інших графіків. Кожен із цих контролерів може завантажувати й інший FXML, утворюючи ієрархію вкладеності. На рис. 3.3 продемонстровано структуру частини проєкту, що відповідає за реалізацію графічного інтерфейсу програмного застосунку:

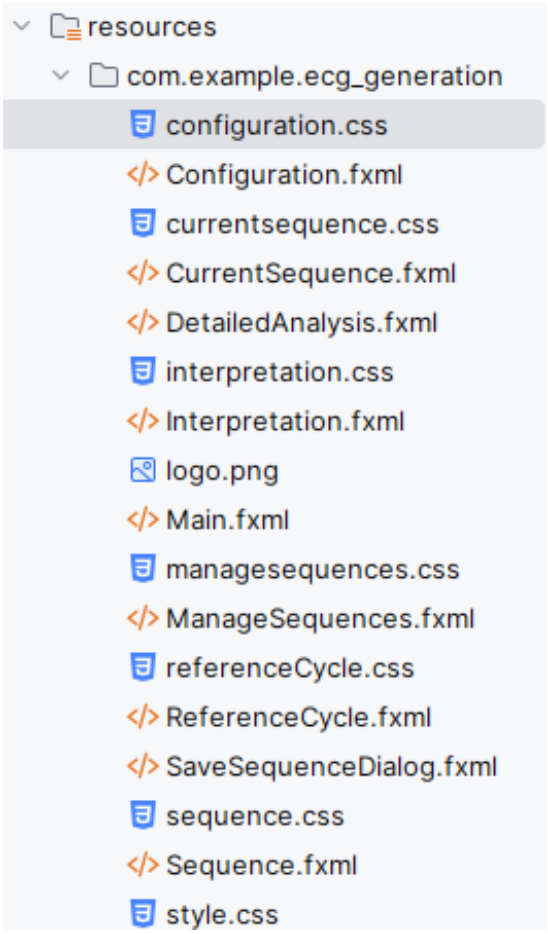

Рисунок 3.3 — Структура частини проєкту з реалізацією користувацького інтерфейсу в IntelliJ IDEA

Стилі описані в окремих CSS: наприклад, configuration.css задає кольори та поведінку кнопок у розділі налаштувань, currentSequence.css – зовнішній вигляд Canvas і лейблів результату TWA. Завдяки цьому, внесення змін до оформлення одного екрану не впливає на інші, що дає нам можливість вносити зміни в існуючий дизайн застосунку без потреби

втручатися в програмну реалізацію.

В процесі реалізації та тестування, такий підхід дозволив нам швидко змінювати окремі частини UI, легко додавати нові екрани та підтримувати єдиний стиль всього застосунку. Завдяки розділенню розмітки й стилів ми можемо окремо зосередитися на логіці контролерів, а для реалізації чи внесення змін в користувацький інтерфейс – в директорію resources. Розглянемо реалізацію кожної сторінки детальніше.

Сторінка 1. Головне вікно (Main Window) є першим екраном, який з'являється при запуску програми. Використовується класом AppLauncher, що завантажує FXML-шаблон головного вікна і створює сцену JavaFX. У центрі розташовані дві великі кнопки: «Configuration» і «Interpretation» (рис. 3.4). Вони об'єднані в рядок із двома ToggleButton всередині ToggleGroup, що гарантує, що обрана може бути лиш одна з вкладок. Ініціалізація контролера MainController налаштовує обробники кліків таким чином, щоб при натисканні на «Configuration» завантажувався підвіконний контейнер з конфігураційними вкладками, а при «Interpretation» — із вкладками для аналізу та інтерпретації сигналів.

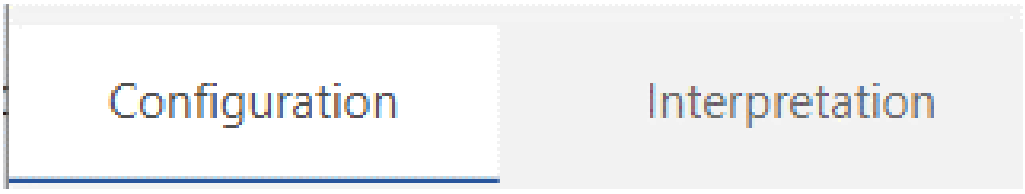

Рисунок 3.4 — Структура FXML контейнера Main

Сторінка 2. Конфігураційний контейнер (Configuration View) являє собою екран-контейнер для двох підвкладок: «Reference Cycle» і «Sequence» (рис. 3.5). FXML-шаблон обгортає панель AnchorPane із двома кнопками для переключення вкладок, куди динамічно підвантажуються два інші FXML-файли. Контролер ConfigurationController у методі initialize() задає групу перемикачів, слухає їх та замінює вміст панелі на відповідний

дочірній корінь.

Рисунок 3.5 — Структура FXML контейнера Configuration та логіка завантаження вкладок

Сторінка 3. Reference Cycle є власне FXML-шаблоном із графіком для формування еталонного ЕКГ-циклу та чотирма слайдерами для кожної із п’яти хвиль P, Q, R, S і T. Контролер ReferenceCycleController опрацьовує рух кожного слайдера — змінює відповідні поля у масиві WaveParameters[], а після кожної зміни викликає метод перемальовки drawSingleCycle() (рис. 3.6). Цю сторінку користувач бачить відразу після вибору вкладки «Reference Cycle» у Configuration View.

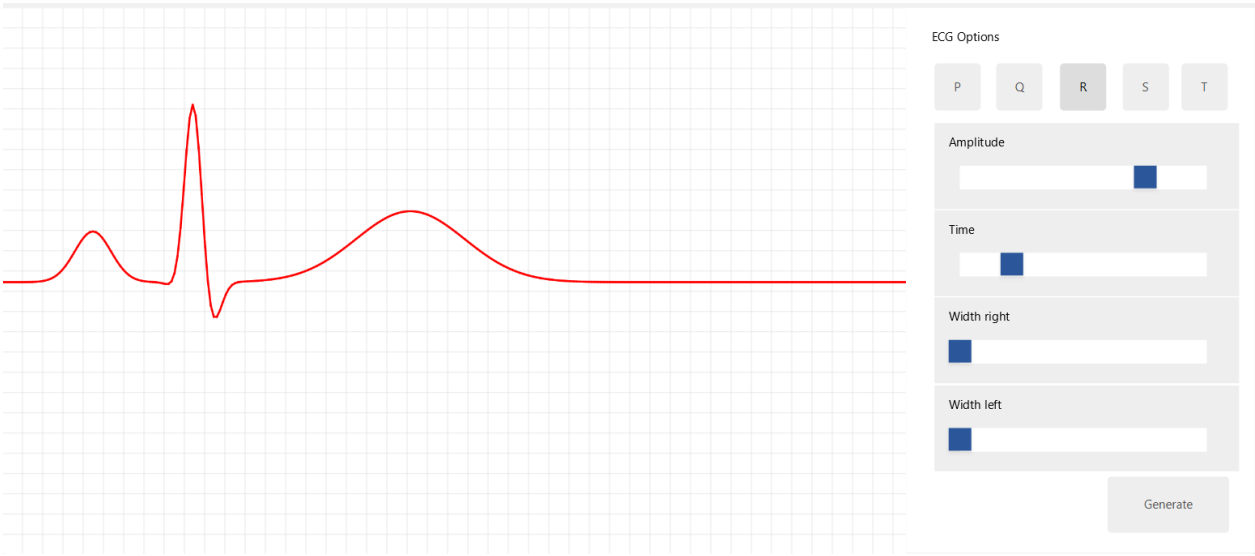

Рисунок 3.6 — Інструмент налаштування еталонного циклу: Canvas і п’ять слайдерів

Сторінка 4. Sequence Generation зустрічає користувача після того, як

еталонний цикл збережено в статичній змінній (рис. 3.7). FXML містить графік, два слайдери (рівень шуму і рівень альтернації зубця T), текстове поле для кількості циклів, кнопки «+»/«–» для коригування циклів, кнопку задання псевдовипадкового розподілу рівня амплітуди кожного окремого зубця T, а також кнопку «Save». Контролер SequenceController, при зміні будь-якого параметра, заново генерує масив сигналу методом generateMultiCycleSignal(), відмальовує його та дає можливість відкрити модальне вікно для збереження сконфігурованої полідовності.

Рисунок 3.7 — Параметри генерації послідовності: шум, цикли, T-альтернація

Сторінка 5. Діалог збереження є окремим FXML-файлом із полем вводу назви, трьома лейблами для відображення параметрів і кнопкою «Confirm» (рис. 3.8). Контролер SaveSequenceDialogController приймає дані з SequenceController, показує їх у діалозі, а при підтвердженні викликає EcgSequenceService для збереження сутності. Після успішної операції діалог закривається.

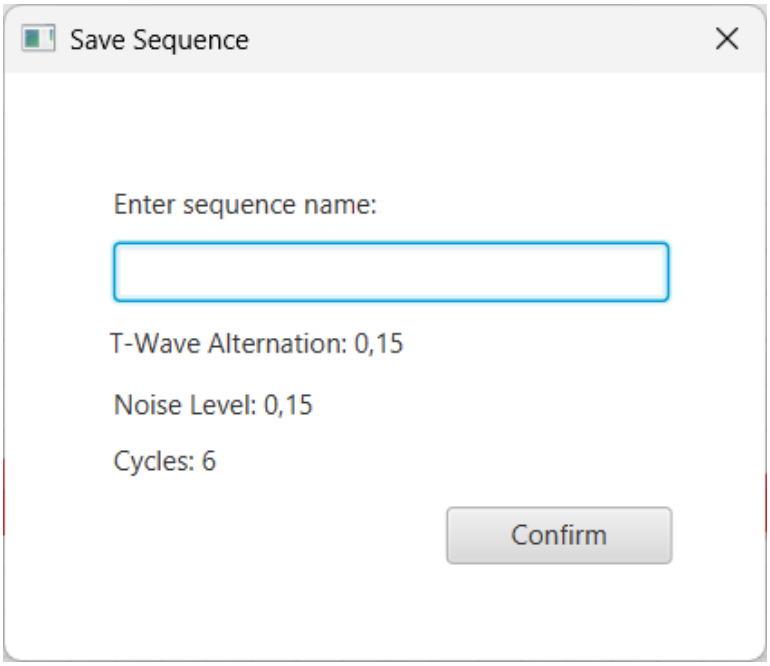

Рисунок 3.8 — Діалог збереження послідовності з полем для назви та параметрами

Сторінка 6. Interpretation View аналогічно до Configuration View, містить дві вкладки: «Manage Sequences» і «Current Sequence» з додатковою кнопкою перемикання на сторінку детального аналізу (рис. 3.9). FXML-шаблон — це AnchorPane з ToggleButton-ами, під яким розташований StackPane. Контролер сторінки підвантажує відповідний дочірній корінь згідно з обраною користувачем кнопкою.

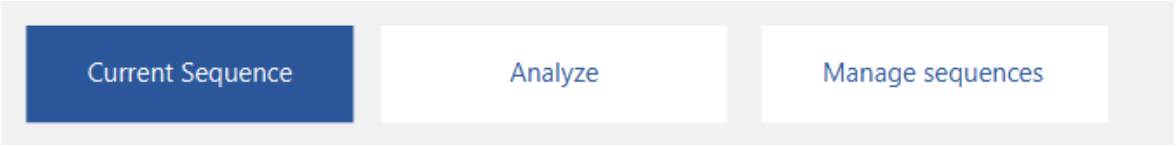

Рисунок 3.9 — Контейнер Interpretation із вкладками керування та аналізу

Сторінка 7 (рис. 3.10) містить таблицю із наступними стовпцями: назва, кількість циклів, рівень заданого користувачем шуму, рівень заданої альтернації, довжина. Кожен рядок містить кнопку «Select» (для переходу до

аналізу), «Rename» та «Delete». Контролер ManageSequencesController при ініціалізації завантажує всі записи через сервіс, заповнює таблицю, а при натисканні «Select» викликає SelectedSequenceHolder.setSelectedSequenceId(id) і перемикає вкладку на «Current Sequence» .

Name	*Cycles*	*T-Wave Alt*	*Noise*	*Length*	*Actions*
1. 1. Контрольна — Без Альтернації, Без Шуму	10	0.0	0.0	3000	Choose Delete
2. Позитивна Альтернація 0.3, Без Шуму	10	0.2977272727272727	0.0	3000	Choose Delete
4. Змішана Альтернація (Т хвилі змінні, з доданим шумом)	10	0.2454545454545454	0.14545454545454548	3000	Choose Delete
5. Високий шум, без альтернації	10	0.0	0.30000000000000004	3000	Choose Delete
6. Позитивна Альтернація + Шум	10	0.29545454545454547	0.19545454545454552	3000	Choose Delete
7. Мінімальна альтернація 0.05, шум 0.0	10	0.05	0.0	3000	Choose Delete
8. 64 цикли, висока альтернація, шум 0.1	64	0.5	0.09545454545454546	19200	Choose Delete

Рисунок 3.10 — Таблиця керування збереженими послідовностями

Сторінка 8. На цьому екрані відображається вибрана послідовність на Canvas (рис. 3.11). Контролер CurrentSequenceController після ініціалізації запитує у сервісу дані вибраної послідовності, малює сигнал. Після завершення обчислень, по натисканню на кнопку MMA: відповідно до отриманого результату обчислень, змінюється колір індикатора (зелений або червоний) і відображається числовий результат у Label.

Рисунок 3.11 — Вікно поточного сигналу з індикатором стану

Сторінка 9 містить три окремі Canvas для відображення послідовності, обраної користувачем, графіка чергування парних/непарних Т зубців, графіка, що відображає рівень альтернації від циклу до циклу по амплітудах зубців Т, а також текстові лейбли для виведення інтерпритацій результатів та оцінки ризику (рис. 3.12). Контролер DetailedAnalysisController при відкритті приймає сигнал, обчислені позиції піків і масив ΔTWA та генерує відповідні графіки.

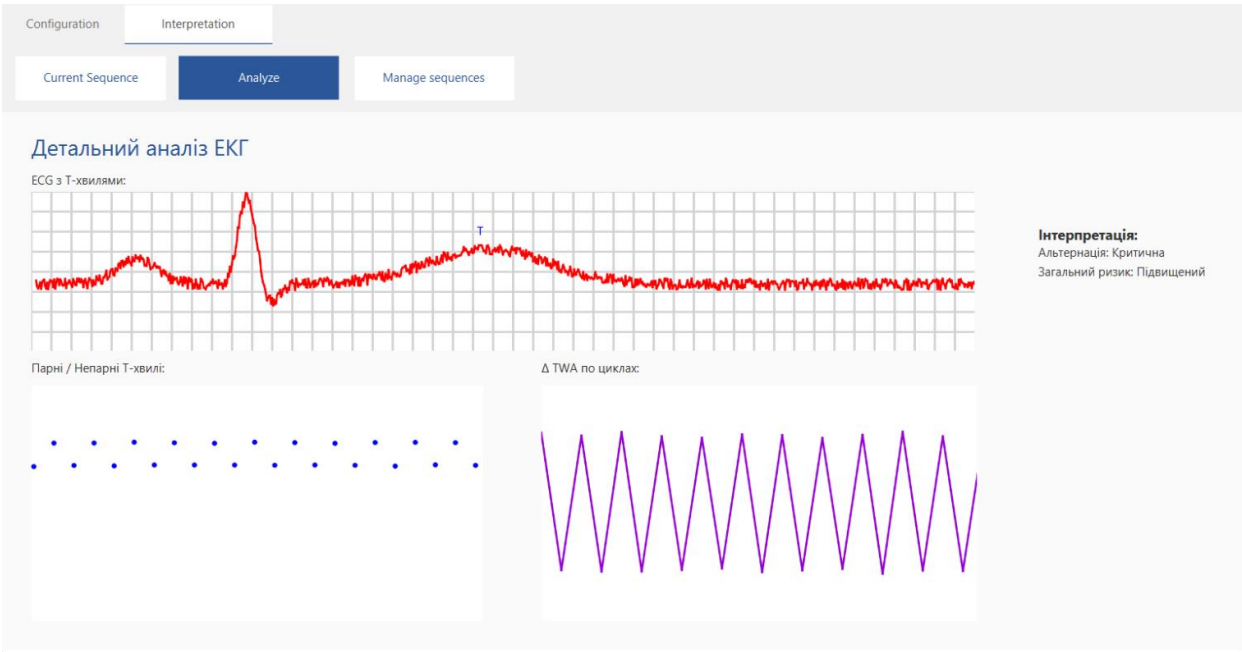

Рисунок 3.12 — Детальний аналіз: графіки та текстові інтерпритації

3.2 UML-діаграми взаємодії користувача з програмним застосунком

У даному підрозділі розглянемо два ключові сценарії користувацької взаємодії з програмним застосунком, кожний з яких ілюструватиметься окремою UML-діаграмою взаємодії. Перший сценарій включатиме етап налаштування еталонного електрокардіографічного циклу, генерації, конфігурації та збереження послідовності в базу даних. Другий сценарій

розглядатиме логіку виброу вже збереженої послідовності та аналізу рівня альтернації зубця Т алгоритмом ММА.

3.2.1 Варіанти використання програмного застосунку

Програмний застосунок, що реалізовується, формується з чотирьох ключових функціональних модулів, які покриватимуть повний цикл роботи з синтетичними ЕКГ-сигналами: налаштування еталонного циклу, генерація послідовності циклів, управління збереженими даними та аналіз рівня Т-альтернації із можливістю отримання поглибленого звіту-дослідження. На рис. 3.13 наведено діаграму варіантів використання програмного застосунку:

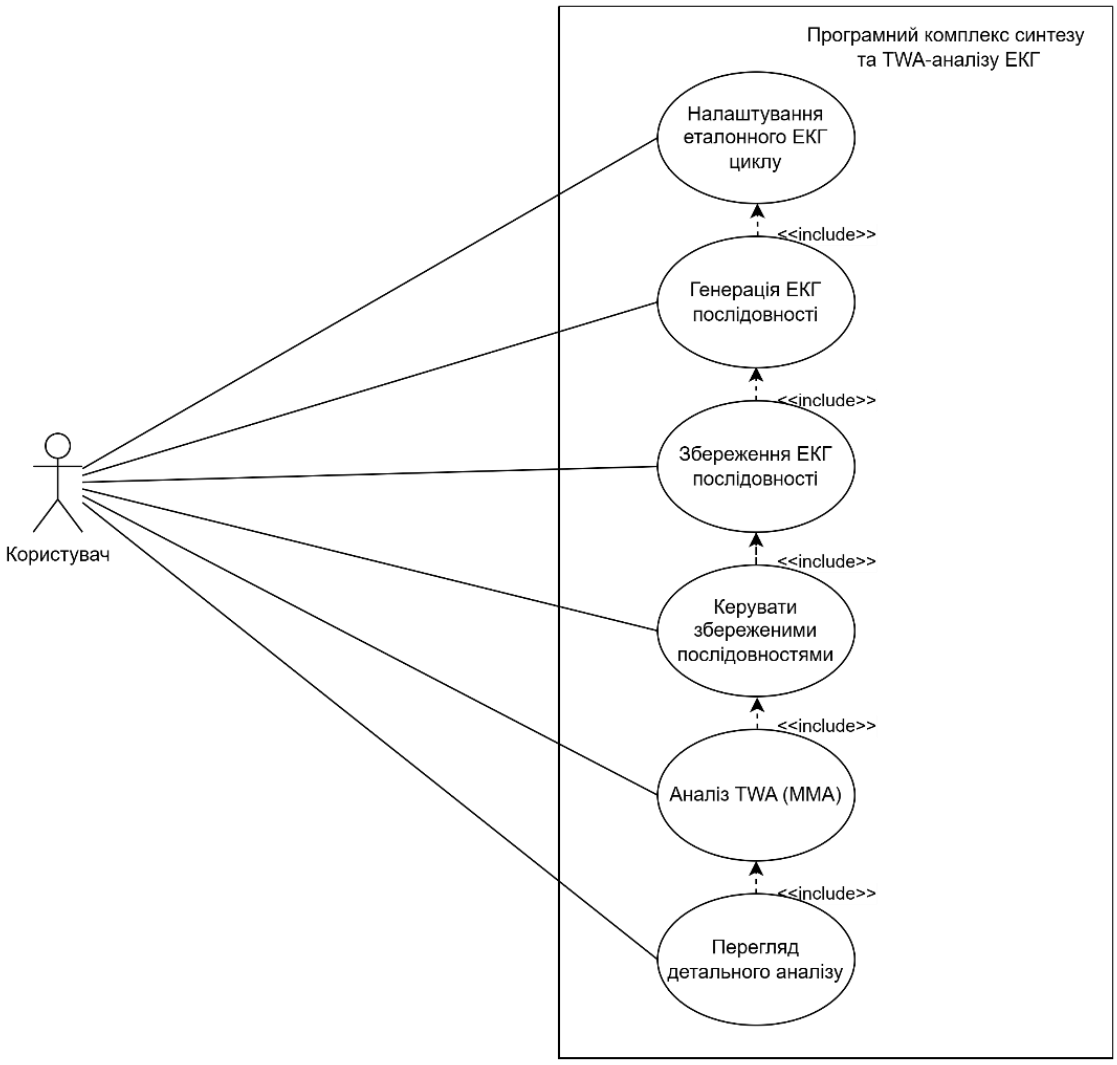

Рисунок 3.13 — Діаграма варіантів використання: ідентифікація ролей та основних модулів генерації, детекції й збереження даних

Перший модуль надає користувачеві інструментарій для формування еталонного серцевого циклу. В ньому реалізовані п’ять пов’язаних між собою слайдерів, які забезпечують можливість задавання аплітуди, часової позиції та двосторонньої ширини кожної з п’яти хвиль P, Q, R, S і T. Результат взаємодії зі слайдерами миттєво відображатиметься на графіку з метою надання можливості користувачеві візуально контролювати форму хвиль та результат конфігурації. Після встановлення всіх параметрів, користувач натискає кнопку переходу до наступного екрана, де задані еталонні значення стають основою для генерації самої послідовності циклів.

Реалізація другого модулю покриває генерацію електрокардіографічної послідовності з урахуванням трьох додаткових параметрів: кількості циклів, рівня шуму та рівня альтернації зубця T в парних циклах. Під час взаємодії з інтерфейсом, користувач має змогу коригувати налаштування, змінювати рівень амплітуди, задавати кількість циклів, коригувати рівень шуму, активувати генерацію псевдовипадкового розподілу рівня амплітуд зубців T, де всі результати конфігурації будуть миттєво відображатись на оновленій послідовності. В модулі міститься кнопка кнопка збереження послідовності, по натисканню на яку, відкривається модальне вікно, в якому користувач має можливість задати назву послідовності. Після користувацького підтвердження, по натискнанню на фінальну кнопку збереження, активується логіка збереження сконфігурованого сигналу (назва, параметри послідовності, масив сигналу) до бази даних.

Третій модуль відповідає за перегляд і управління збереженими записами. В ньому реалізована таблиця, в якій відображаються всі попередньо збережені послідовності з основними атрибутами: назвою, кількістю циклів, рівнями шуму, рівнями альтернації, довжиною сигналу, тощо. Кожен рядок таблиці містить кнопку «Обрати» для подальшого аналізу, а також поле для зміни назви послідовності та кнопку видалення. Подія натискання кнопки видалення запускає програмний процес pre-destroy

валідації шляхом відкриття відповідного модального вікна, в якому користувач має підтвердити свій вибір.

Після вибору конкретної послідовності для аналізу, користувач буде переходити до центрального аналітичного модуля, де одним натисканням кнопки, застосовуватиметься модифікований алгоритм ковзнього середнього (MMA - Modified Moving Average Beat Analysis) для обчислення показника рівня альтернації зубця T [45] . В цьому модулі реалізований індикатор, який по виконанню аналізу послідовності алгоритмом, змінюватиме колір маркеру стану: зелений позначатиме відсутність альтернації, червоний сигналізуватиме про її наявність. Відповідні результати на індикаторі доповнюватимуться конкретним відображенням рівня альтернації, що був визначений алгоритмом.

Наслідком забезпечення високого рівня узгодженості вищерозглянутих модулів, формується зрозумілий користувацький шлях, від створення еталонного циклу до збереження, селекції, швидкого оцінювання рівня альтернації та глибшої аналітичної інтерпритації. Розглянуті сценарії використання та поділ на функціональні сервіси слугують основою архітектури програмного застосунку та її реалізації в програмному коді.

3.2.2 Генерація еталонного циклу та збереження послідовності

Розглядаємо перший із двох ключових сценаріїв користувацької взаємодії з нашим програмним комплексом – налаштування еталонного електрокардіоциклу, його перетворення на послідовність ЕКГ-сигналів та збереження отриманих даних у локальну базу PostgreSQL. Сценарій покриватиме весь шлях від первинної конфігурації параметрів еталонного циклу до надійної персистенції результатів, демонструючи, як кожен архітектурний шар забезпечує чітку розмежованість обов’язків, виконуючи при цьому визначені функції. На рис. 3.14 наведено діаграму взаємодії користувача з програмним застосунком при генерації еталонного кардіоцику та збереження синтетичної послідовності.

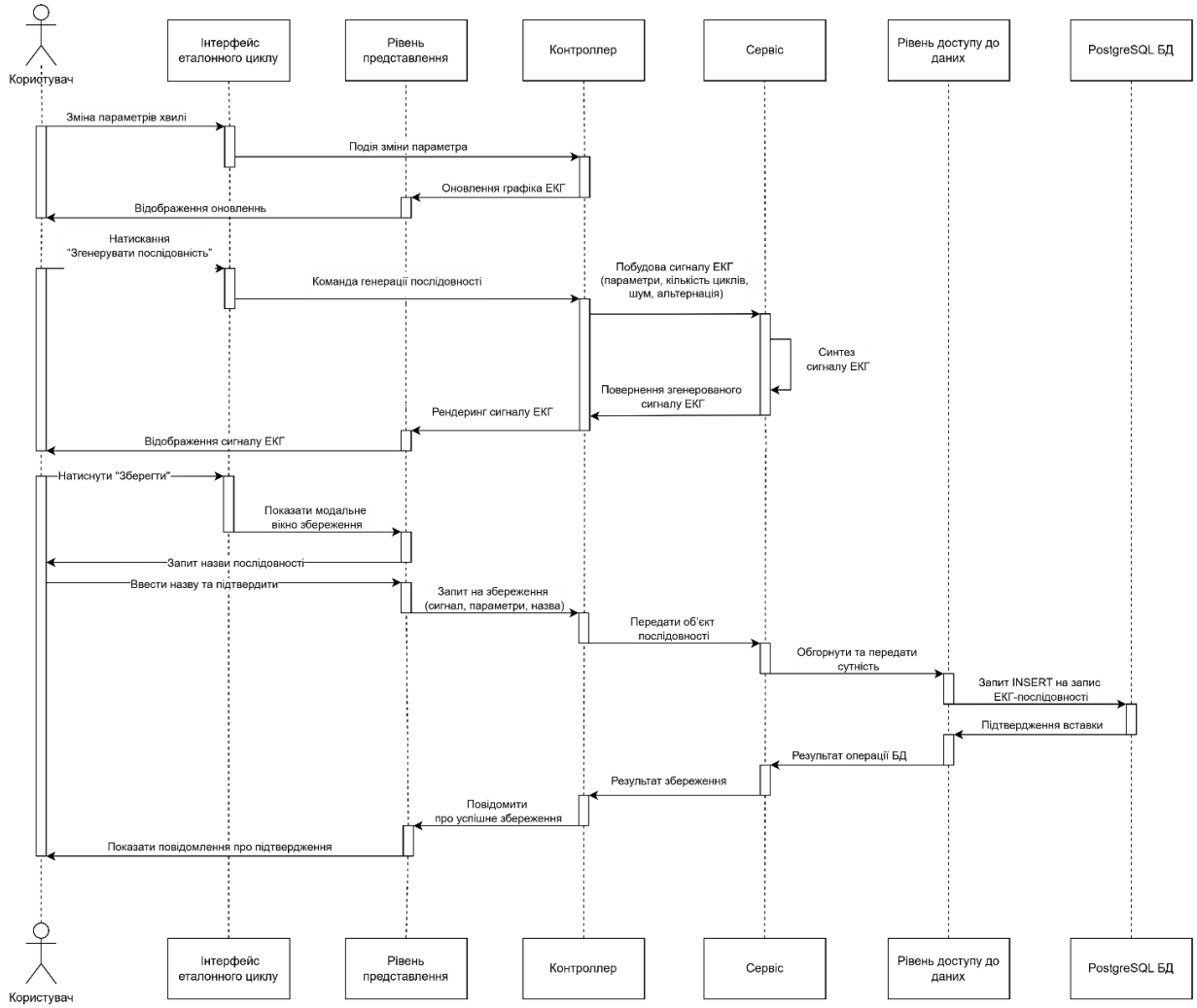

Рисунок 3.14 — Послідовність взаємодії при генерації еталонного циклу та збереженні синтетичної послідовності

Взаємодія користувача з інтерфейсом починається з конфігурування еталонного кардіоциклу за допомогою інтуїтивно зрозумілих слайдерів у вікні конфігурації, де задається амплітуда, часова позиція та симетрія п’яти хвиль: P, Q, R, S та T. Кожна зміна одного з параметрів миттєво провокує роботу рівня доступу до даних, контролер реагує на подію зміни слайдера, оновлює внутрішню модель параметрів і дає команду на перемальовку графіку (Canvas), на якому власне відмальовується цикл. Користувач отримує миттєвий візуальний зворотний зв’язок, що дозволяє йому коригувати форму еталонного циклу до бажаного вигляду.

Як тільки процес конфігурації завершується, користувач натискає

кнопку «Generate Sequence». Ця дія ініціює виклик обробнику подій SequenceController, який збирає всі параметри (кількість циклів, рівні шуму та альтернації, назву та номер точок) і передає їх на обробку до шару бізнес-логіки. Компонент синтезу сигналу обробляє масив значень ЕКГ, відтворюючи еталонний цикл задану кількість разів із заданими спотвореннями. Згенерований сигнал повертається у рівень доступу до даних, де миттєво відображається на екрані у вигляді кривої.

Наступним кроком користувач натискає кнопку «Save». Контролер відкриває модальне вікно діалогу збереження, запитуючи назву послідовності. Після введення і підтвердження назви SequenceController формує запит до сервісу EcgSequenceService, передаючи йому як параметри конфігурації, так і власне масив синтетичних даних. Сервіс створює об’єкт-сутність із цими даними та звертається до механізму JPA - EntityManager, який конертує сутність на SQL-команду INSERT та записує її у таблицю ecg_sequences локальної бази PostgreSQL.

Після успішного завершення транзакції сервіс повертає результат контролеру, який закриває діалог і повідомляє користувача про успішне збереження. На цьому етапі завершується ланцюжок взаємодії. Від інтуїтивного налаштування еталонного кардіоциклу до надійного збереження складного бінарного масиву ЕКГ-даних у базі, із чітким розподілом обов’язків між рівнем доступу до даних, рівнем з бізнес-логікою та рівнем відображення.

3.2.3 Вибір послідовності та аналіз Т-альтернації

Другим ключовим сценарієм користувацької взаємодії із системою є вибір попередньо збереженої синтетичної послідовності ЕКГ та оцінка рівня альтернації зубця за допомогою модифікованого алгоритму ковзнього середнього (Modified Moving Average). На рис. 3.15 наведено діаграму взаємодії користувача з програмним застосунком під час підвантаження екг-послідовності та здійснення аналізу послідовності на предмет альтернації зубця Т.

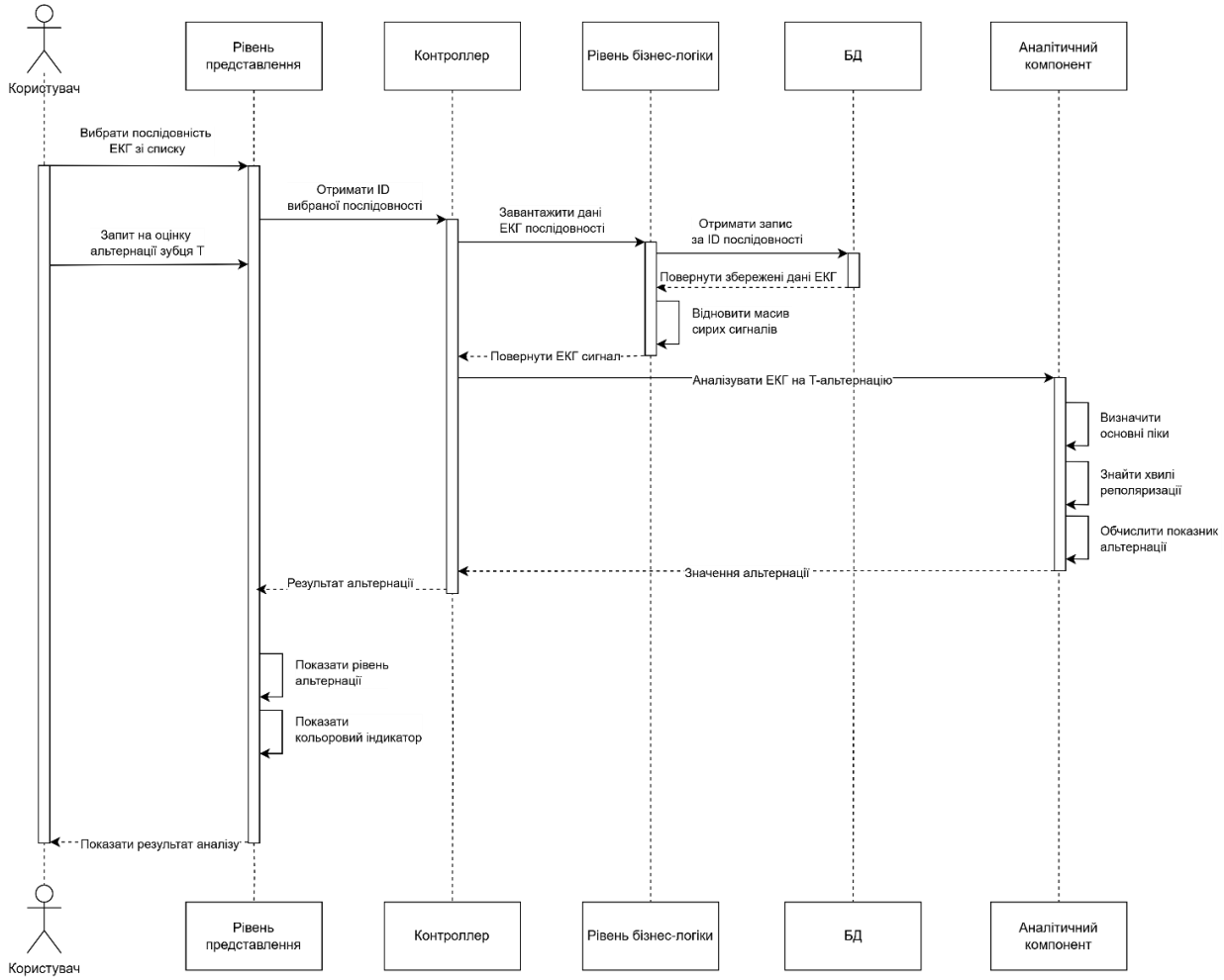

Рисунок 3.15 — Послідовність взаємодії під час вибору послідовності та аналізу Т-альтернації

Приступаючи до процесу аналізу, користувач у вікні керування збереженими записами обирає потрібну послідовність зі списку. Інтерфейс фіксує цю дію та активує компонент Current Sequence UI. Контролер цієї вкладки звертається до спільного сховища SelectedSequenceHolder, щоб дізнатися ідентифікатор обраного запису, а потім викликає сервіс EcgSequenceService для завантаження повного масиву сигналу з бази даних. Сервіс, у свою чергу, через ORM-шар JPA ініціює запит в базу даних та отримує збережений бінарний масив. Після цього він реконструює масив чисел із бінарного представлення та повертає його контролеру. Контроллер передає масив сигнала до аналітичного компонента MMAAnalyzer, який

послідовно виконує виявлення R-піків як еталонних точок початку серцевого циклу, локалізацію Т-хвиль у реполяризаційному сегменті, обчислення різниці середніх амплітуд парних і непарних Т-хвиль та формування єдиного числового показника альтернації.

Після отримання значення рівня альтернації, контролер порівнює його з встановленим порогом. Якщо метрика перевищує межу, рівень відображення демонструє користувачу червоний індикатор і текстове повідомлення про виявлену альтернацію; в іншому випадку — зелене світло та повідомлення про відсутність значимих для нас коливань. Одночасно користувач бачить точне числове значення рівня альтернації на панелі результатів.

Сценарій ілюструє чітке розмежування обов’язків, де рівень відображення відповідає за взаємодію і візуалізацію, контроллер координує потік даних, рівень бізнес-логіки відповідає за доступ до сховища та реконструкцію сигналу, рівень доступу до даних забезпечує зберігання й доставку бінарних даних, а Analysis Component виконує суто обчислювальну роботу.

3.3 Робота із базою даних

У сучасних десктопних застосунках, які оперують великими обсягами наукових чи медичних даних, завдання надійного та ефетктивного збереження результатів експерементів та обчислень є фунаментально важливим. В контексті даної роботи, для вирішення цієї задачі, було обрано модель об’єктно-реляційного відображення на основі JPA (Jakarta Persistence API) із реалізацією Hibernate та реляційну СУБД PostgreSQL з метою уможливлення роботи з сутностями баз даних, як Java-об’єктами, забезпечуючи автоматичну трансформацію цих об’єктів в SQL-операції під капотом, без потреби реалізовувати складні запити мануально [47].

Для розуміння застосованих підходів, детальніше розглянемо

технології, імплементовані в наш додаток для взаємодії з базою даних на різному рівні абстракцій. Найвищим рівнем абстракції в цьому випадку є Java Persistence API стандартизований специфікаційний інтерфейс у Java, який визначає, як об’єкти у додатку можуть бути пов’язані з даними у реляційній базі. JPA як стандарт, не виконує збереження чи читання даних, а лиш задає так званий контракт, угоду про те, як має виглядати ця взамодія: набір анотацій та інтерфейсів, які диверсифікують і спрощують процес об’єктно-реляційного відображення (ORM) [48]. Ми позначаємо Java-класи як сутності, а атрибути цих класів — як стовпці таблиць бази даних, описуючи це за допомогою анотацій. Далі за роботу з цими сутностями відповідає реалізація JPA, в нашому випадку, як реалізацію стандарту, ми використовуватимемо Hibernate. Останній є open-source реалізацією JPA, що додає власні механізми оптимізації та розширені можливості накшталт кешування першого рівня, батч-обробку SQL-запитів та динамічний генератор SQL [49]. Коли бізнес-логіці потрібно зберігати або вичитувати об’єкт, вона викликає методи EntityManager (напр. persist(), find(), createQuery()), а Hibernate в свою чергу підбирає оптимальне SQL-формулювання, готує через Java Database Connectivity (JDBC) відповідні запити і надсилає їх у базу даних. На найнижчому рівні цієї взаємодії, працюватиме JDBC, який є по суті низькорівнеим стандартом для роботи з реляційними базами даних в Java [50]. Після того як Hibernate сформував SQL-запит, він використовує JDBC-драйвер для встановлення мережевого або локального з’єднання з сервером PostgreSQL, підготовки PreparedStatement, підстановки параметрів і виклику executeQuery() чи executeUpdate(). Сервер бази даних обробляє запит, виконує операцію і повертає результат у вигляді ResultSet, який за допомогою JDBC-API перетворюється на потік рядків. Hibernate отримує цей потік і картографує кожен рядок на відповідної сутності, які в програмному коді реалізуються у вигляді об’єктів [51].

В застосунку, після запуску програми, відбувається завантаження

конфігурацій Hibernate з ресурсу persistence.xml, де зазначено ім’я Persistence Unit, драйвер PostgreSQL, адресу з’єднання та властивості пулу з’єднань. Після цього, створюється єдиний екземпляр EntityManagerFactory, який вбудовує в себе пул JDBC-з’єднань і метадані про всі сутності [52]. Ця фабрика відповідає за життєвий цикл з’єднань із базою даних і слугує своєрідною точкою входу для ініціалізації об’єктів EntityManager, кожен із яких використовується в рамках окремої транзакції. На рис. 3.16 наведено діаграму життєвого циклу транзакції з сутністю EcgSequence:

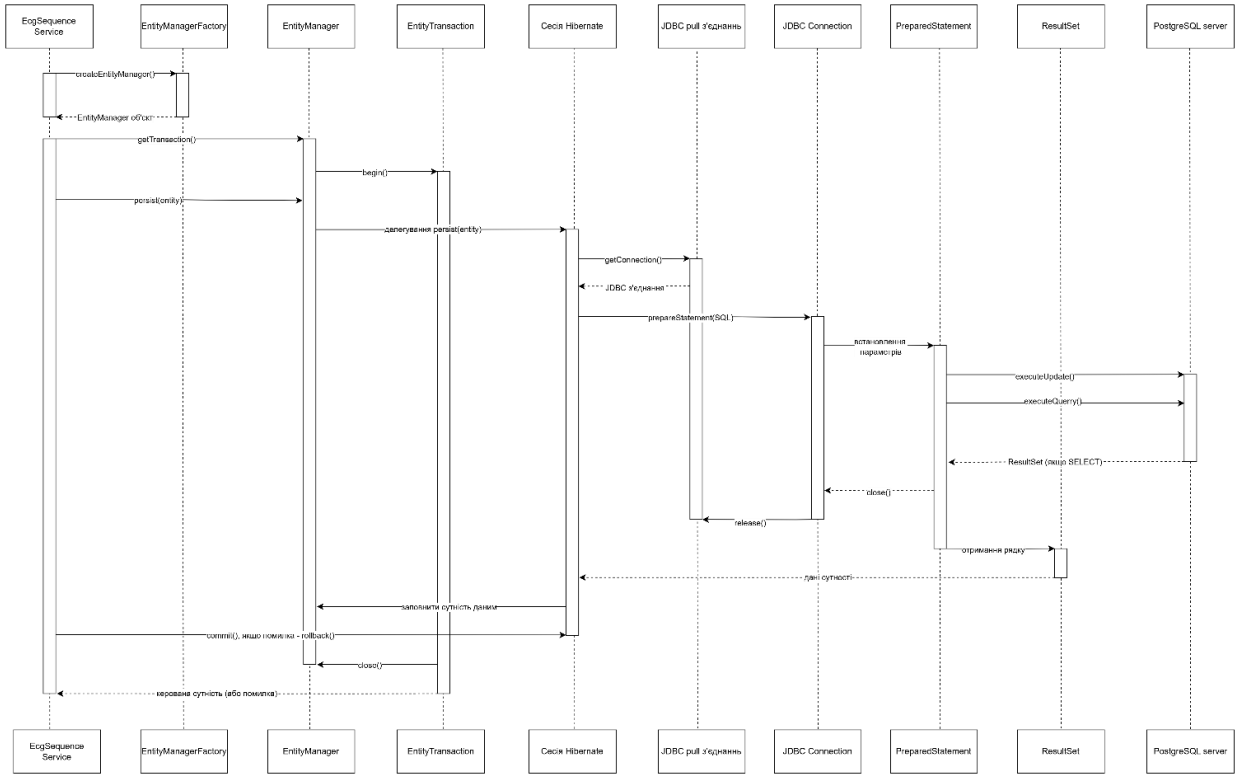

Рисунок 3.16 — Життєвий цикл транзакції з EcgSequence

У нашій моделі, головна сутність – клас EcgSequence (рис. 3.17) – позначатиметься анотацією @Entity та відображатиметься на таблиці ecg_sequences. Поле signalData, яке зберігає масив подвійних значень ЕКГ, позначатиметься аннотацією @Lob, що автоматично перетворюватиме його в байтове поле PostgreSQL. Інші атрибути сутності, такі як назва послідовності, кількість циклів, рівень шуму та альтернації – відображатимуться як

стандартні текстові та числові колонки. Під час виклику шаром бізнес-логіки методу збереження, Hibernate створює екземпляр EntityManager та починає транзакцію. Спершу він аналізує об'єкт сутності та генерує SQL-запит INSERT (рис. 3.19). Далі, через JDBC-драйвер, запит відсилається до сервера PostgreSQL, який фізично записує дані на диск, а після успішного виконання транзакція фіксується.

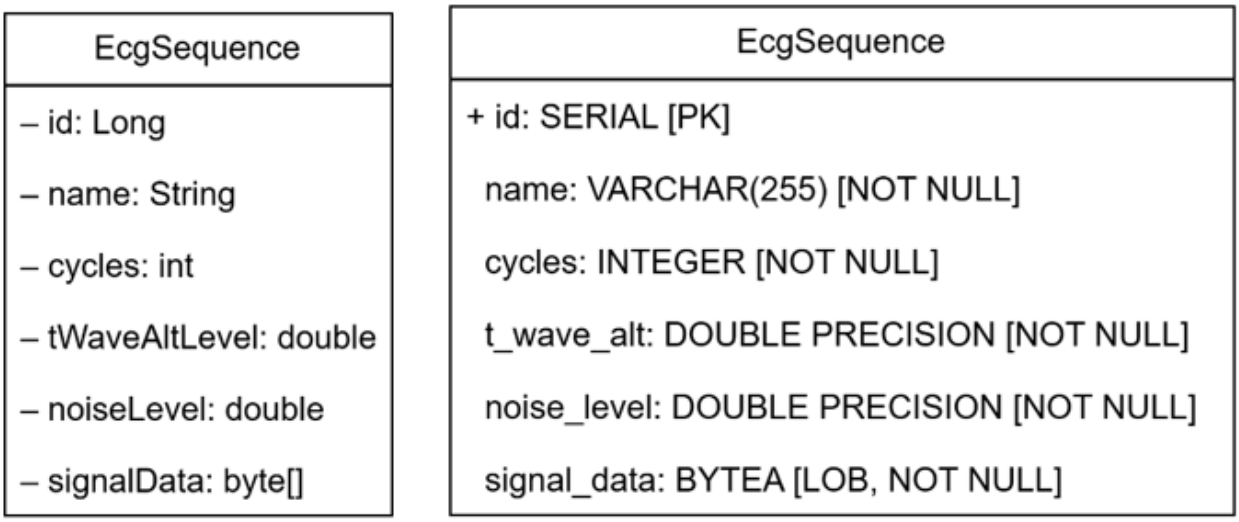

Рисунок 3.17 — Структурна модель сутності EcgSequence

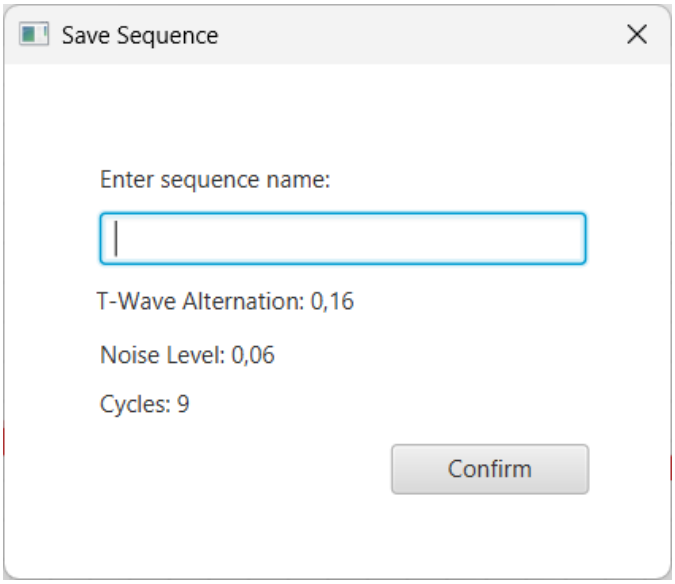

Рисунок 3.18 — Знімок з екрану реалізованого вікна SaveSequenceDialogController

```
Hibernate:
    insert
    into
        ecg_sequences
        (cycles, name, noiseLevel, signalData, tWaveAltLevel)
    values
        (?, ?, ?, ?, ?)
=== Saved to DB ===
Name: Sequence no 1
Cycles: 9
T-Wave Alt: 0.1609090909090908
Noise: 0.0599999965320934
Signal size: 9000
====================
```

Рисунок 3.19 — Запит insert, згенерований Hibernate на збереження сконфігурованої послідовності до бази даних

Зчитування даних в нашому застосунку відбуватиметься аналогічно, але в зворотному порядку: бізнес-шар надсилає до сервісу запит на отримання запису за ідентифікатором, Hibernate формує JPQL-запит, перетворює його на SQL SELECT і передає через JDBC. Сервер СУБД виконує пошук у таблиці та повертає результати у вигляді рядків. Hibernate знову картографує кожен рядок на новий екземпляр EcgSequence.

Реалізоване середовище дозволить нам легко підміняти реалізацію DAO-класів на мок-об’єкти, масштабувати рішення, змінювати СУБД або налаштування пулу з’єднань без зміни суті програмної логіки, що відповідає перевинному баченню стосовно масштабування програмного застосунку в подальших дослідженнях.

3.4 Методика виявлення альтернації Т-хвилі

В цьому підрозділі, з бажанням провести валідації алгоритму та формування висновків про ефективність роботи застосунку, здійснимо демонстрацію наскрізного тесту —«генерування → збереження → аналіз» — для трьох різних сценаріїв конфігурації контрольних вибірок:

- послідовність A з рівнем альтернації T-хвилі α = 0.25 мВ,
- послідовність B з рівнем альтернації T-хвилі α = 0.01 мВ.
- послідовність C з заданим випадковим рівнем аплітуд зубця T з метою перевіркою роботи алгоритму з різним рівнем амплітуди зубця проте відсутньою закономірністю у чергуванні, яка характерна сигналам з альтернацією

Мета тестування — ствердити: алгоритм Modified Moving Average (MMA) достовірно розрізняє «патологічний» і «нормальний» сигнали, а всі шари застосунку коректно взаємодіють між собою.

Крок 1. генерація послідовності A (α = 0.25 мВ). У вікні Sequence Generation (рис. 3.20) задаємо:

- Кількість циклів= 30,
- Рівень шуму = 0.03 мВ,
- Рівень альтернації = 0.25 мВ.

Canvas відтворює виразне чергування амплітуд T-хвилі.

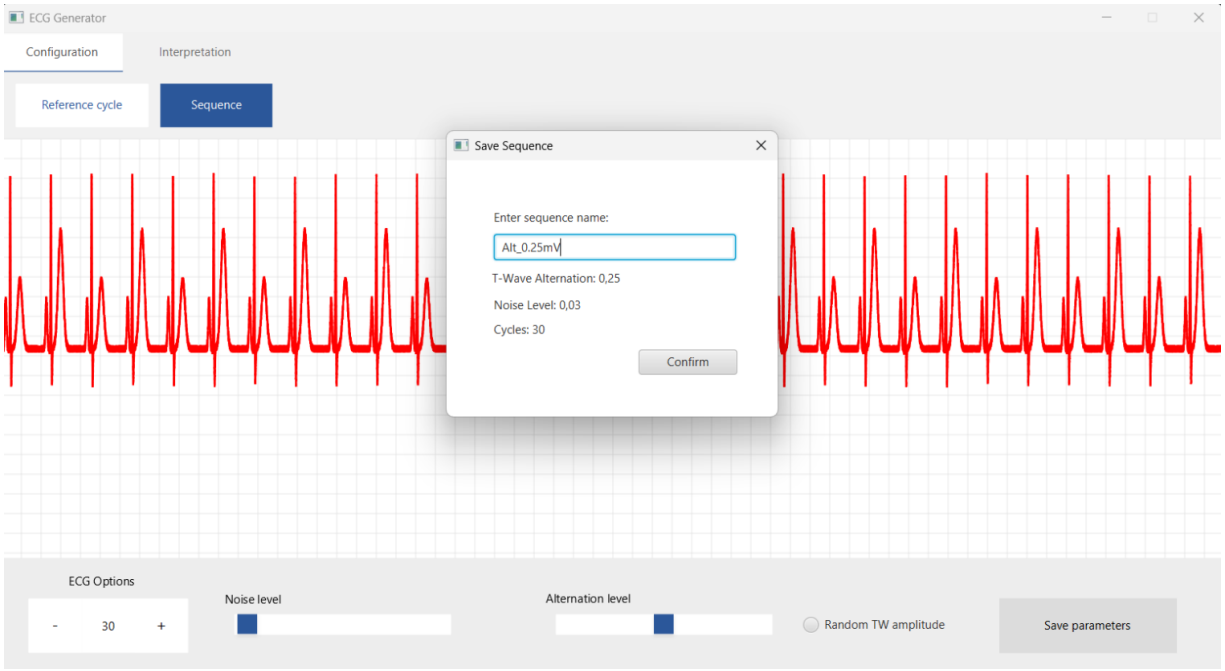

Рисунок 3.20 — Послідовність A із чергуванням T-хвиль

Натискаємо Save parameters, вводимо назву Alt_0.25mV,

підтверджуємо. Hibernate формує SQL-вставку до таблиці сутності ecg_sequences, що фіксується в логах (рис. 3.21):

```
Hibernate:
    insert
    into
        ecg_sequences
        (cycles, name, noiseLevel, signalData, tWaveAltLevel)
    values
        (?, ?, ?, ?, ?)
=== Saved to DB ===
Name: Alt_0.25mV
Cycles: 30
T-Wave Alt: 0.2481818077780983
Noise: 0.02909090909090909
Signal size: 30000
====================
```

Рисунок 3.21 — Діалог збереження та SQL-INSERT

Крок 2 –генерація послідовності B ($\alpha = 0.01$ мВ). Без перезавантаження сцени, зміщуємо слайдер альтернації до 0.01 мВ; решта параметрів лишаються незмінними. Чергування Т-хвиль майже непомітне. Зберігаємо під назвою Alt_0.01mV (рис. 3.22).

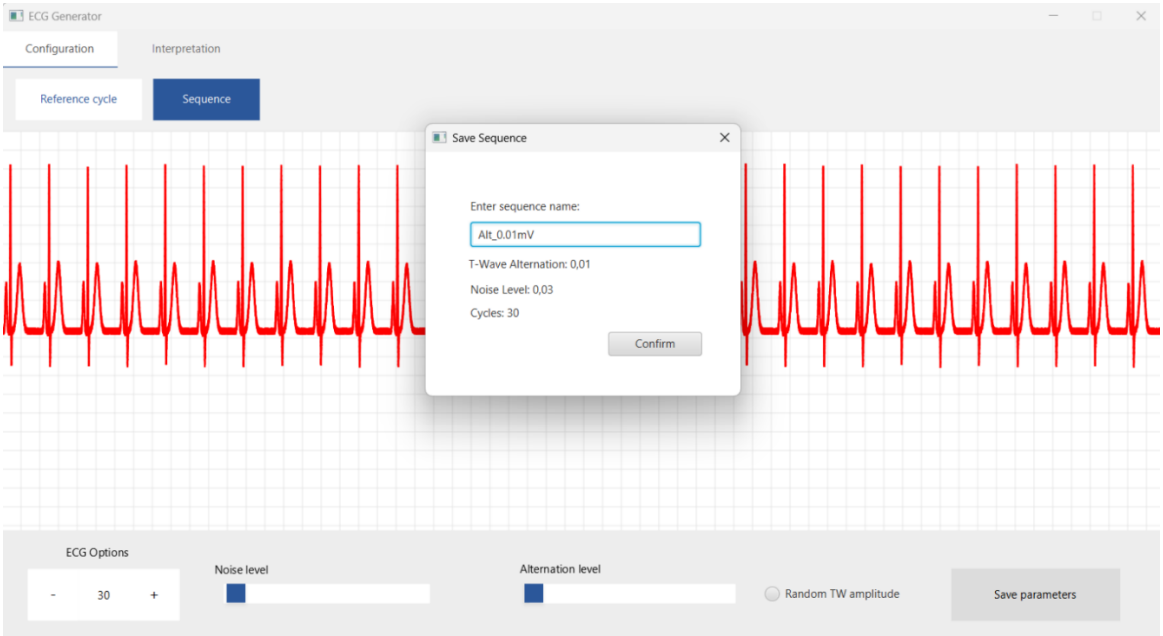

Рисунок 3.22 — Послідовність B із відсутньою альтернацією

Крок 3 – генерація послідовності C (α =random). Для створення послідовності з псевдовипадковим розподілом рівня амплітуд зубців T, натискаємо відповідну кнопку Random TW amplitude альтернації до 0.01 мВ; решта параметрів лишаються незмінними. Зберігаємо під назвою Alt_0.01mV (рис. 3.23).

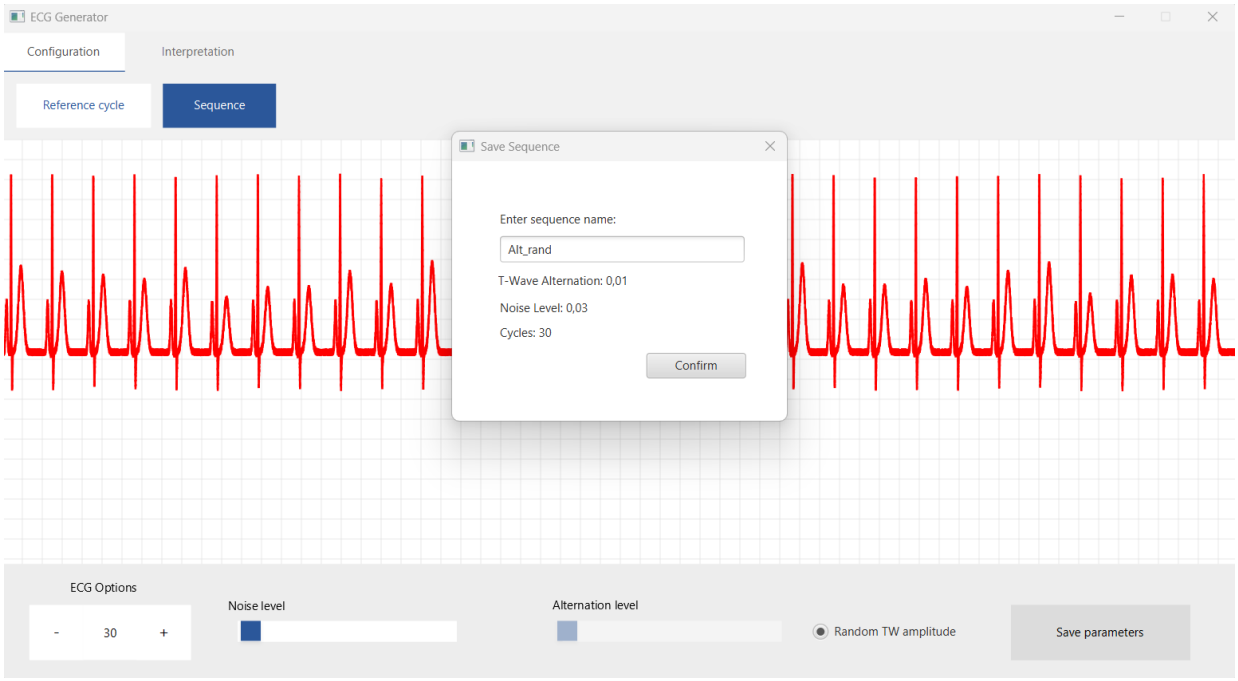

Рисунок 3.23 — Послідовність C із псевдовипадковою альтернацією

Крок 4 – здійснюємо аналіз послідовності A. Переходимо до Interpretation → Manage Sequences, натискаємо Chose у рядку Alt_0.25mV; вікно Current Sequence оновлюється графіком сигнало-кривої.

Натискаємо кнопку MMA. Компонент MMAAnalyzer:

1. Шукає R-піки;
2. локалізує T-хвилі після кожного R;
3. аналізує регулярність чергувань;
4. обчислює рівень альтернації за (3.1);

$$\Delta = |\overline{T_{even}} - \overline{T_{odd}}| \qquad (3.1)$$

5. порівнює з порогом $\theta = 0.02$ мВ і присвоює клас Positive alternans.

Індикатор стану світиться червоним, під Canvas виведено повідомлення $\Delta TWA > 0.02$ мВ → TWA DETECTED (рис. 3.24).

Рисунок 3.24 — Результат аналізу послідовності A

Крок 5 – аналіз послідовності B. Тим самим шляхом обираємо Alt_0.01mV і натискаємо кнопку MMA. MMA відображає:

$$\Delta TWA < 0.02 \text{ мВ} \rightarrow \text{TWA ABSENT} \qquad (3.2)$$

Індикатор стає зеленим, текст повідомляє про відсутність значущої альтернації (рис. 3.25).

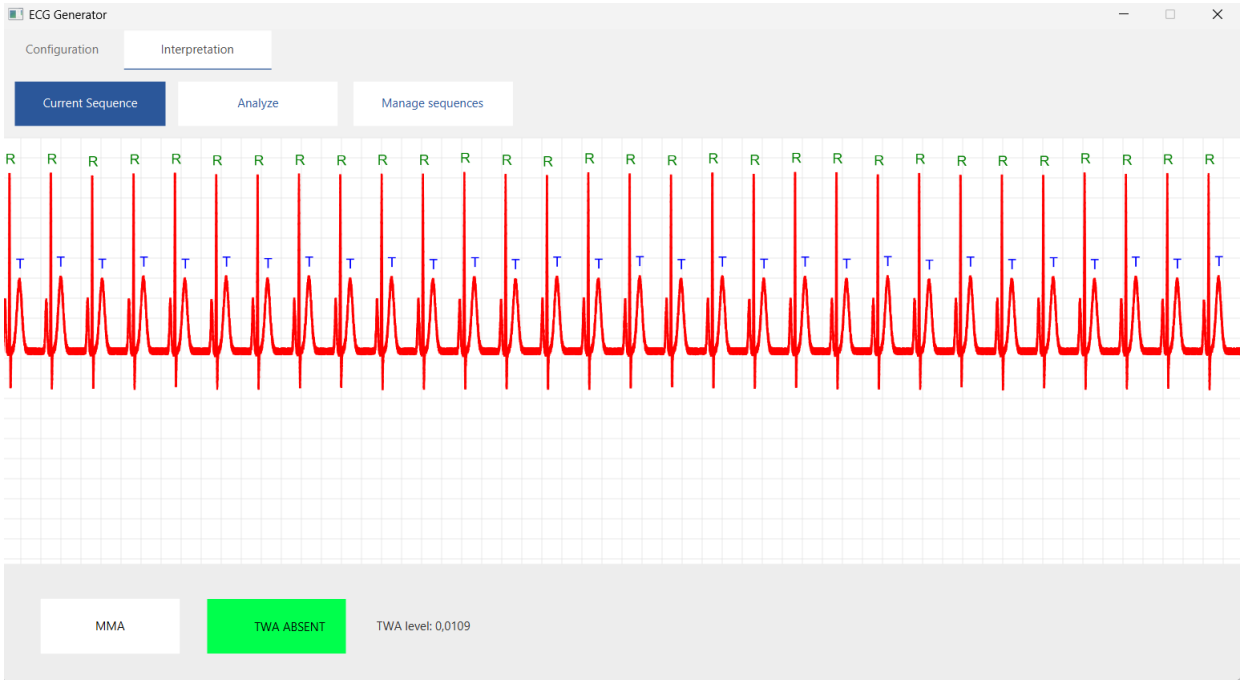

Рисунок 3.25 — Результат аналізу послідовності B — зелений індикатор

Крок 6 – аналіз послідовності C. Обираємо Alt_rand і запускаємо кнопку MMA. Активується процедура препроцесингу сигналу. Алгоритм аналізує частоту та регулярність зміни знаків рівнів аплітуд та виявляючи відсутність закономірного чергування, класифікує відсутність альтернації на послідовності (рис. 3.26).

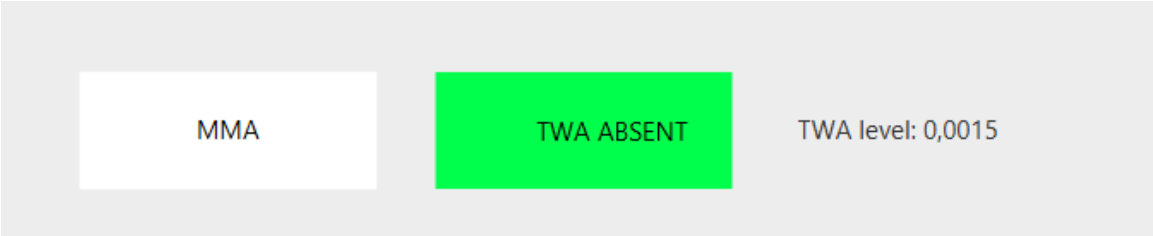

Рисунок 3.26 — Результат аналізу послідовності C — зелений індикатор

Крок 7 – Перевірка операцій CRUD. У вкладці Manage Sequences послідовність B перейменовуємо на Alt_low і одразу видаляємо тестовий запис. У консольному логові Hibernate видно SQL-команди UPDATE, а

таблиця оновлюється без перезапуску інтерфейсу (рис. 3.27).

```
Hibernate:
    select
        es1_0.id,
        es1_0.cycles,
        es1_0.name,
        es1_0.noiseLevel,
        es1_0.signalData,
        es1_0.tWaveAltLevel
    from
        ecg_sequences es1_0
    where
        es1_0.id=?
Hibernate:
    update
        ecg_sequences
    set
        cycles=?,
        name=?,
        noiseLevel=?,
        signalData=?,
        tWaveAltLevel=?
    where
        id=?
```

Рисунок 3.27 — Логи Hibernate під час перейменування/видалення

Модуль генераці сигналів створює контрольовані послідовності з різним рівнем альтернації зубця Т. Ріень доступу до даних (JPA → Hibernate → PostgreSQL) надійно зберігає та відтворює великі бінарні масиви, підтримує повний набір реалізованих CRUD-операцій. Алгоритм ММА чітко відрізняє патологічну та фізіологічну реполяризацію, повертаючи стабільні показники, демонструючи коректну роботу в умовах відмінності рівня амплітуд зубців, що аналізуються, враховуючи закономірності

відповідних чергувань.

Візуальний модуль забезпечує миттєвий зворотний зв’язок, де користувач отримує числову оцінку рівня альтернації та кольорову індикацію праткично миттєво.

3.5 Розрахунок економічного ефекту за темою дипломної роботи

У межах дипломної роботи розглянуто кілька альтернативних підходів до реалізації програмного засобу для генерації й аналізу ЕКГ-сигналів із виявленням альтернації зубця Т. Зокрема оцінювалися три варіанти комбінацій функцій:

F_1 — середовище та мова програмування; F_2 — метод математичної генерації синтетичного сигналу; F_3 — алгоритм виявлення альтернації. Для кожної функції запропоновано по три можливі реалізації (А, Б, В). У результаті сформовано дев’ять потенційних варіантів (рис. 3.28), серед яких до детального аналізу відібрано два найперспективніші за попередньою експертною оцінкою.

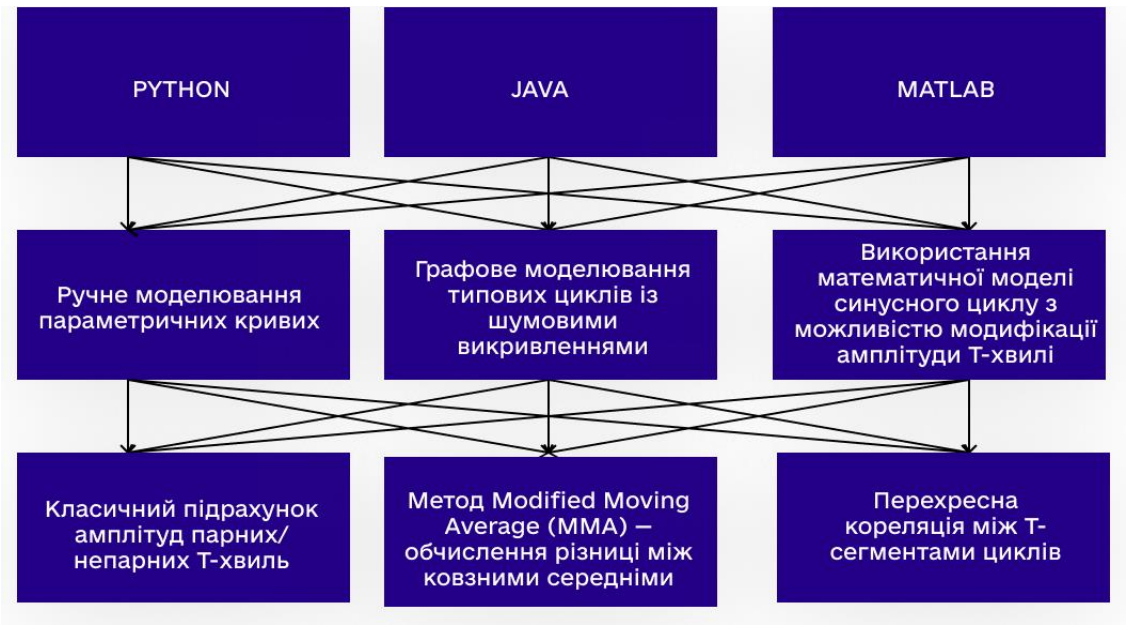

Рисунок 3.28 – Морфологічна карта відображає можливі комбінації реалізації кожної з вищенаведених функцій

З метою оцінки раціональності застосування тих чи інших технічних рішень, використовується метод функціонально-вартісного аналізу (ФВА), який дозволить нам встановити відповідність між функціональними характеристиками та відповідними витратами на розробку цих характеристик, впровадження та супровід.

ФВА дозволить нам виконати формалізацію основних функцій розробленого програмного засобу, ідентифікувати альтернативні шляхи щодо реалізації кожної функції, побудувати систему критеріїв оцінювання та визначити вагомості параметрів якості, досягнувши досить оптимального співвідношення «витрати / функціональна цінність».

Функціонально-вартісний аналіз (ФВА) є методикою зіставлення корисного ефекту кожної функції із витратами на її реалізацію та експлуатацію. Для оцінки застосунку було сформовано систему параметрів: X_1 — обсяг коду, X_2 — обчислювальна складність генератора, X_3 — тривалість розробки, X_4 — точність алгоритму виявлення альтернації. Кожному параметру присвоєно вагу шляхом попарного порівняння, після чого розраховано інтегральні коефіцієнти технічної якості для двох фінальних варіантів.

За результатами ФВА та порівняння витрат перевагу отримав варіант, що передбачає:

- Мову програмування Java
- Гнучке моделювання ЕКГ, можливість зміни амплітуди, шуму, кількості циклів
- Модифікований алгоритм ковзнього середнього Modified Moving Average (MMA) для виявлення рівня альтернації зубця Т.

Така конфігурація продемонструвала найвище співвідношення технічної якості до витрат і забезпечує оптимальну собівартість реалізації.

Відтак, саме цей варіант рекомендовано прийняти як базовий для подальшого впровадження й використання в програмному застосунку.

Вартість розробки програмного продукту становить 259 368,22 грн.

3.6 Безпекова модель та запобігання ненавмисному видаленню

В програмному застосунку безпекову модель та захист даних реалізовано на низці взаємопов'язаних рівнів. На етапі доступу до бази даних, всі CRUD операції виконуються через ORM зв'язки JPA/Hibernate, а отже SQL запити генеруються автоматично та передаються до PostgreSQL у вигляді Prepared Statement. Параметризація запитів унеможливлює ін'єкцію шкідливих команд, оскільки значення ідентифікаторів, назв і будь-яких числових полів ніколи не вбудовуються в текст SQL у відкриту, а передаються через заздалегідь скомпільовані плейсхолдери [53]. Додатково Hibernate використовує вбудовану валідацію анотацій (@NotNull, @Size тощо), що блокує спроби записати в таблицю ecg_sequences дані, які не відповідають названим обмеженням схеми.

Сам сервер PostgreSQL розгортається з розмежуванням ролей: окремий користувач програмного застосунку має лише команди SELECT, INSERT, UPDATE, DELETE у конкретній схемі і не володіє правами створення функцій або доступу до системних таблиць [54]. З'єднання JDBC встановлюється з ввімкненим SSL [55], що шифрує дані в каналі та захищає паролі від прослуховування у локальній мережі. Конфігураційний файл persistence.xml зберігається поза папкою resources, а реальні паролі передаються через змінні оточення під час запуску, що позбавляє необхідності тримати секрети у репозиторії коду.

Ще одним важливим етапом інформаційного захисту є реалізований механізм запобігання ненавмисній втраті даних. Перед видаленням будь-якої послідовності, екран Manage Sequences ініціює подвійну валідацію: спершу відображається діалог із деталями запису (назва, кількість циклів, рівень шуму, альтернація) і кнопками «Confirm»/«Cancel»; при підтвердженні у базі розгортається транзакція з рівнем ізоляції READ COMMITTED. Якщо під час

підтвердження у паралельній сесії обраний запис уже змінено або видалено, Hibernate піднімає виняток OptimisticLockException, транзакцію відкочує, а користувач отримує повідомлення про конфлікт, отже битих записів не виникає.

Висновок до розділу 3

Побудована програмна система відповідає всім визначеним функціональним та нефункціональним вимогам, демонструє цілісну, добре масштабовану архітектуру. Декомпозиція інтерфейсу на FXML-шаблони, окремі CSS-стилі та контролери забезпечила чіткий розподіл обов'язків, що суттєво спростило як розробку, так і подальше тестування та візуальне налаштування компонентів. Ієрархічна навігаційна модель із динамічним підвантаженням вкладених екранів дозволила об'єднати уніфікований користувацький досвід із можливістю незалежного розвитку кожного підмодуля, від конфігурування еталонного циклу до аналітичних панелей глибокого аналізу. Проведене наскрізне тестування «генерація → збереження → аналіз» підтвердило коректність взаємодії усіх шарів, продемонструвавши ефективну та якісну роботу всіх програмних компонетів. Контроллери гарантовано передають дані бізнес-логіці, сервісний рівень безпомилково інтегрується з JPA/Hibernate, а ORM-шар, в свою чергу, забезпечує транзакційну цілісність операцій у PostgreSQL. Імплементований алгоритм Modified Moving Average довів здатність достовірно розрізняти патологічні та фізіологічні реполяризаційні процеси на синтетичних сигналах різного рівня складності, що дає нам підстави зробити висновок щодо правильності математичної моделі та адекватності її програмної реалізації. Реалізовані безпекові механізми мінімізують ризики втрати даних і роблять систему стійкою до типових атак за рахунок реалізації параметризованих запитів, SSL-шифрування, рольового розмежування прав, подвійної валідації перед видаленням послідовності з бази, потенційно руйнівних дій і оптимістичного

блокування транзакцій. Економічна оцінка, виконана методами функціонально-вартісного аналізу, підтвердила, що обрана комбінація технологій (Java + Hibernate + PostgreSQL + MMA) забезпечує найкраще співвідношення технічної якості до витрат. Практична реалізація програмного застосунку доводить життєздатність концепції, демонструє високу ефективність розроблених рішень і створює надійну основу для подальших досліджень і впровадження в реальних клінічних або дослідницьких середовищах.

ЗАГАЛЬНІ ВИСНОВКИ

У підсумку виконаної роботи, сформовано цілісний, обґрунтований підхід до визначення предиктора раптової серцевої смерті, альтернації зубця Т. Спираючись на аналіз актуальних джерел, уточнено сучасні дефініції РСС, виявлено її домінантні етіологічні групи та показано, що електрофізіологічні аномалії формують електричний субстрат, здатний перерости у фатальні аритмії. В роботі було здійснено:

1. Детальний порівняльний огляд QT-залежних маркерів, синдрому Бругада й альтернації зубця Т, який продемонстрував, що альтернація зубця Т вирізняється поєднанням високої чутливості до ранньої реполяризаційної нестабільності та можливості реєстрації під час звичайного холтерівського моніторингу, що обґрунтовує її вибір як основного програмного застосунку для моніторингу
2. Через відсутність у відкритому доступі достатньо репрезентативних ЕКГ з вираженою альтернацією, інтегровано гаусівську генеративну модель породження ЕКГ-циклу з контрольованим рівнем шуму й амплітуди альтернації, в також алгоритмічний модуль MMA, оптимізований для виявлення цієї альтернації.
3. Створено програмний застосунок з використанням технологій Java / JavaFX / JPA-Hibernate-PostgreSQL, що забезпечує чітке розмежування презентаційного, сервісного та персистентного шарів, спрощує модульне тестування й масштабування та гарантує крос-платформенність завдяки fat-JAR-дистрибутиву. Застосунок дає користувачеві послідовний сценарій «конфігурація - генерація - збереження - аналіз»: інтерактивні слайдери дозволяють формувати еталонний цикл, засіб генерації

- формувати великі набори синтетичних сигналів, ORM-шар — надійно зберігати бінарні дані й метадані, обчислювальний модуль — повертати числову оцінку рівня альтернації з інтерпретацією ризику.

4. Проведено тестуваннят застосунку, яке показало, що при альтернації 0,25 мВ алгоритм демонструє впевнене позитивне спрацювання, тоді як при 0,01 мВ та випадкових амплітудах, не виявляє ознак, характерних для явища альтернації, що підтверджує коректність математичної моделі й програмної реалізації.
5. Інтегровано безпекову модель із застосуванням SSL-шифрування, параметризованих запитів до бази та рольового розмежування й оптимістичного блокування транзакцій
6. Проведено функціонально-вартісний аналіз, що засвідчив економічну доцільність обраного набору технологій та підходів, визначивши собівартість розробки в 259 тис. грн при найвищому показнику якості серед альтернатив.

Результати роботи демонструють ефективність моделювання ЕКГ із керованим рівнем альтернації у поєднанні з адаптованим алгоритмом MMA для визначення предиктору PCC, надаючи дієвий інструмент, що може застосовуватись дослідниками та клініцистами, здатний прискорити верифікацію нових методик аналізу та слугувати навчальним середовищем для підготовки фахівців з електрокардіографії.

Розроблений застосунок формує практичне підґрунтя для подальшого розширення — від інтеграції з реальними холтерівськими потоками і машинного навчання до розгортання клієнт-серверної системи з дистанційним моніторингом пацієнтів, відкриваючи перспективу створення повноцінної програмної платформи раннього попередження раптової серцевої смерті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Асоціація кардіологів України. Лікування шлуночкових порушень серця та профілактика раптової серцевої смерті : рекомендації / уклад. : О. С. Сичов (модератор), О. В. Коркушко, В. О. Бобров [та ін.] ; за участю В. Г. Дзяка, В. М. Коваленка. Київ : Асоціація кардіологів України, 2009. 56 с.

2) Серцево-судинні захворювання : класифікація, стандарти діагностики та лікування / Асоціація кардіологів України ; за ред. В. М. Коваленка, М. І. Лутая, Ю. М. Сіренка, О. С. Сичова. 2-ге вид., переробл. та доповн. Київ : МОРІОН, 2016. 192 с. ISBN 978-966-2066-62-3.

3) Рекомендації 2015 Європейського товариства кардіологів (ESC) щодо лікування пацієнтів з шлуночковими аритміями та профілактики раптової серцевої смерті [Електронний ресурс]. Health-ua. Режим доступу: https://health-ua.com/article/4964-rekomendatc-2015-vropejskogo-tovaristva-kardologv-ESC-shodo-lkuvannya-patcn (дата звернення: 07.05.2025).

4) Chugh S. S., Jui J., Gunson K., Stecker E. C., John B. T., Thompson B. [та ін.] Current burden of sudden cardiac death: multiple source surveillance versus retrospective death certificate-based review in a large U.S. community. J. Am. Coll. Cardiol. 2004. Vol. 44, No. 6. P. 1268–1275. DOI: 10.1016/j.jacc.2004.06.029.

5) Chugh S. S., Reinier K., Teodorescu C., Evanado A., Kehr E., Al Samara M., Mariani R., Gunson K., Jui J. Epidemiology of sudden cardiac death: clinical and research implications // Progress in Cardiovascular Diseases. — 2008. — Vol. 51, No. 3. — P. 213–228. — DOI: 10.1016/j.pcad.2008.06.003.

6) Myerburg R. J., Junttila M. J. Sudden cardiac death caused by coronary heart disease. Circulation. 2012. Vol. 125, No. 8. P. 1043–1052. DOI:

10.1161/CIRCULATIONAHA.111.023846. Режим доступу: https://www.ahajournals.org/doi/full/10.1161/CIRCULATIONAHA.111.023846 (дата звернення: 12.05.2025).

7) Гострі порушення серцевого ритму [Електронний ресурс]. Тернопільський державний медичний університет. Режим доступу: https://fm.tdmu.edu.ua/materiali-dla-pidgotovki/interni/medicina-nevidkladnih-staniv/7-gostri-porusenna-sercevogo-ritmu (дата звернення: 11.05.2025).

8) Chandrasekaran B., Kurbaan A. S. Myocardial Ischemia [Електронний ресурс]. StatPearls [Internet]. Treasure Island (FL) : StatPearls Publishing, 2024. Режим доступу: https://www.ncbi.nlm.nih.gov/books/NBK537076/ (дата звернення: 11.05.2025).

9) Постінфарктне ремоделювання міокарда [Електронний ресурс]. Медицина світу. Режим доступу: http://msvitu.com/archive/2013/august/article-3.php (дата звернення: 17.05.2025).

10) Особливості постінфарктного ремоделювання серця в пацієнтів із гострим інфарктом міокарда [PDF, Електронний ресурс]. Терапевтичний вісник. 2011. Режим доступу: https://terapevtyka.com.ua/index.php/journal/article/download/73/55/ (дата звернення: 11.05.2025).

11) Ар'я А., Хаг'ю М., Садр Амелі М. А. Ризик стратифікації для аритмічної смерті після інфаркту міокарда : сучасний погляд та майбутній вектор розвитку. International Journal of Cardiology. 2006. Vol. 108, No. 2. С. 155–164. DOI: 10.1016/j.ijcard.2005.05.011.

12) Dispersion of repolarization between myocardial cells [Електронний ресурс]. ResearchGate. Режим доступу: https://www.researchgate.net/figure/Dispersion-of-repolarization-between-

myocardial-cells-which-is-thought-to-provide-the_fig10_277131125 (дата звернення: 19.03.2025).

13) Sarquella-Brugada G., Campuzano O., Arbelo E. [та ін.] Brugada syndrome : clinical and genetic findings. Genetics in Medicine. 2016. Vol. 18, No. 1. P. 3–12. DOI: 10.1038/gim.2015.35.

14) Cutler M. J., Rosenbaum D. S. Explaining the clinical manifestations of T wave alternans in patients at risk for sudden cardiac death. Heart Rhythm. 2009. Vol. 6, Suppl. 3. P. S22–S28. DOI: 10.1016/j.hrthm.2008.10.007.

15) Verrier R. L., Klingenheben T., Malik M. [та ін.] Microvolt T wave alternans : physiological basis, methods of measurement and clinical utility : consensus guideline by the International Society for Holter and Noninvasive Electrocardiology. Journal of the American College of Cardiology. 2011. Vol. 58, No. 13. P. 1309–1324. DOI: 10.1016/j.jacc.2011.06.029.

16) Електричні альтернанти : типи, причини, діагностика та лікування [Електронний ресурс]. Режим доступу: https://clincasequest.academy/electrical-alternans/ (дата звернення: 15.03.2025).

17) Verrier R. L., Kumar K., Nearing B. D. Basis for sudden cardiac death prediction by T wave alternans from an integrative physiology perspective. Heart Rhythm. 2009. Vol. 6, No. 3. P. 416–422. DOI: 10.1016/j.hrthm.2008.11.019.

18) Qu Z., Weiss J. N. Electrophysiological mechanisms underlying T wave alternans and their role in arrhythmogenesis. Frontiers in Physiology. 2021. Vol. 12. Art. 614946. Режим доступу: https://pmc.ncbi.nlm.nih.gov/articles/PMC7969788/ (дата звернення: 12.04.2025).

19) Beeler G. W., Reuter H. Reconstruction of the action potential of ventricular myocardial fibres. Journal of Physiology. 1977. Vol. 268, No. 1. P. 177–210. DOI: 10.1113/jphysiol.1977.sp011853.

20) Luo C. H., Rudy Y. A model of the ventricular cardiac action potential : depolarization, repolarization, and their interaction. Circulation Research. 1991. Vol. 68, No. 6. P. 1501–1526. DOI: 10.1161/01.RES.68.6.1501.

21) Файнзільберг Л. С. Методи та системи штучного інтелекту : підручник. Київ : 7БЦ, 2023. 320 с. С. 65–90. ISBN 978-617-549-255-0.

22) Архітектура та проектування програмного забезпечення : опорний конспект лекцій [Електронний ресурс] / Вінницький нац. техн. ун-т. Режим доступу: http://dspace.wunu.edu.ua/bitstream/316497/24194/1/опорний%20конспект%20лекцій.pdf (дата звернення: 02.04.2025).

23) Муляр В. П. Розробка JavaFX додатків із використанням Scene Builder. Комп'ютерно-інтегровані технології: освіта, наука, виробництво. 2019. Вип. 3. С. 45–51.

24) Архітектура програмного забезпечення: оптимізація коду [Електронний ресурс]. Wezom.com.ua. Режим доступу: https://wezom.com.ua/ua/blog/arhitektura-programnogo-zabezpechennya (дата звернення: 26.03.2025).

25) Особливості переходу на Hibernate 6 : покращення та зворотна сумісність [Електронний ресурс]. DOU.ua. Режим доступу: https://dou.ua/forums/topic/38570/ (дата звернення: 18.04.2025).

26) Java Design Patterns. Data Access Object Pattern in Java [Електронний ресурс]. Режим доступу: https://java-design-patterns.com/patterns/data-access-object/ (дата звернення: 11.03.2025).

27) Гексагональна архітектура в Java : принципи та практики [Електронний ресурс]. Foxminded.ua. Режим доступу: https://foxminded.ua/heksahonalna-arkhitektura-java/ (дата звернення: 14.04.2025).

28) Архітектура мови програмування Java [Електронний ресурс]. Promoter.net.ua. Режим доступу: https://promoter.net.ua/articles/arxitektura-movi-programuvannia-java.html (дата звернення: 30.03.2025).

29) Компіляція та виконання Java застосунків під капотом [Електронний ресурс]. JavaRush. 2023. Режим доступу: https://javarush.com/ua/groups/posts/4044-kompiljacija-i-ispolnenie-java-prilozheniy-pod-kapotom (дата звернення: 20.03.2025).

30) Oracle Java SE Support Roadmap [Електронний ресурс]. Oracle Corp., 2024. Режим доступу: https://www.oracle.com/java/technologies/java-se-support-roadmap.html (дата звернення: 12.03.2025).

31) JavaFX Tutorial : Part 4 — Styling with CSS [Електронний ресурс]. Makery.ch. 2024. Режим доступу: https://code.makery.ch/uk/library/javafx-tutorial/part4/ (дата звернення: 06.04.2025).

32) Jakarta Persistence Query Language [Електронний ресурс]. Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Jakarta_Persistence_Query_Language (дата звернення: 01.05.2025).

33) Hibernate PostgreSQL Database Tutorial [Електронний ресурс]. Java Guides. 2024. Режим доступу: https://www.javaguides.net/2024/05/hibernate-postgresql-database-tutorial.html (дата звернення: 29.04.2025).

34) Архітектура програмного забезпечення [Електронний ресурс]. Вікіпедія. Режим доступу:

https://uk.wikipedia.org/wiki/Архітектура_програмного_забезпечення (дата звернення: 10.04.2025).

35) Application (JavaFX 8) : API Documentation [Електронний ресурс]. Oracle Corp. Режим доступу: https://docs.oracle.com/javase/8/javafx/api/javafx/application/Application.html (дата звернення: 17.04.2025).

36) CRUD операції : створення, читання, оновлення, видалення [Електронний ресурс]. JavaRush. Режим доступу: https://javarush.com/ua/quests/lectures/ua.javarush.web.core.lecture.level22.lecture05 (дата звернення: 25.03.2025).

37) Java Singleton Design Pattern : best practices with examples [Електронний ресурс]. DigitalOcean. 2024. Режим доступу: https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples (дата звернення: 27.03.2025).

38) Learn JPA & Hibernate Series [Електронний ресурс]. Baeldung. 2025. Режим доступу: https://www.baeldung.com/learn-jpa-hibernate (дата звернення: 08.04.2025).

39) How JPA EntityManager handles Hibernate transaction management [Електронний ресурс]. TheServerSide. 2020. Режим доступу: https://www.theserverside.com/video/How-JPA-EntityManager-handles-Hibernate-transaction-management (дата звернення: 31.03.2025).

40) Package javafx.animation : JavaFX API Documentation [Електронний ресурс]. 2025. Режим доступу: https://download.java.net/java/early_access/javafx25/docs/api/javafx.graphics/javafx/animation/package-summary.html (дата звернення: 22.04.2025).

41) Deploying JavaFX Applications [Електронний ресурс]. Oracle Help Center. 2013. Режим доступу: https://docs.oracle.com/javafx/2/deployment/jfxpub-deployment.pdf (дата звернення: 18.03.2025).

42) Creating Fat JARs for JavaFX Applications [Електронний ресурс]. FXApps. 2020. Режим доступу: http://fxapps.blogspot.com/2020/11/creating-fat-jars-for-javafx.html (дата звернення: 13.04.2025).

43) Configure PostgreSQL to allow remote connection [Електронний ресурс]. BigBinary. 2016. Режим доступу: https://www.bigbinary.com/blog/configure-postgresql-to-allow-remote-connection (дата звернення: 30.04.2025).

44) Creating Executables for JavaFX Applications [Електронний ресурс]. Foojay.io. 2023. Режим доступу: https://foojay.io/today/creating-executables-for-javafx-applications/ (дата звернення: 14.04.2025).

45) Файнзільберг Л. С. Комп'ютерна діагностика за фазовим портретом електрокардіограми : монографія. Київ : Освіта України, 2013. 192 с. ISBN 978-966-188-336-8.

46) Рухоме середнє [Електронний ресурс]. Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Рухоме_середнє (дата звернення: 11.04.2025).

47) Mihalcea V. How to map a PostgreSQL HStore entity property with JPA and Hibernate [Електронний ресурс]. 2023. Режим доступу: https://vladmihalcea.com/map-postgresql-hstore-jpa-entity-property-hibernate/ (дата звернення: 01.04.2025).

48) Java Persistence API [Електронний ресурс]. Вікіпедія. Режим доступу: https://uk.wikipedia.org/wiki/Java_Persistence_API (дата звернення: 17.03.2025).

49) Що таке Hibernate Framework в Java та як ним користуватися? [Електронний ресурс]. Highload.tech. 2023. Режим доступу: https://highload.tech/uk/shho-take-hibernate-framework-v-java-ta-yak-nim-koristuvatisya/ (дата звернення: 05.04.2025).

50) Java JDBC API Documentation [Електронний ресурс]. Oracle Corp. 2018. Режим доступу: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/ (дата звернення: 19.04.2025).

51) Janssen T. Hibernate with PostgreSQL — 6 things you need to know [Електронний ресурс]. 2023. Режим доступу: https://thorben-janssen.com/hibernate-postgresql-5-things-need-know/ (дата звернення: 20.04.2025).

52) Hibernate and PostgreSQL configuration using persistence.xml and EntityManager [Електронний ресурс]. JVMHub. 2014. Режим доступу: http://jvmhub.com/2014/10/25/hibernate-and-postgresql-configuration-using-persistence-xml-and-entitymanager/ (дата звернення: 09.04.2025).

53) SQL-ін’єкції : що це, як працює, які типи існують, небезпечність [Електронний ресурс]. Foxminded.ua. 2024. Режим доступу: https://foxminded.ua/sql-iniektsii/ (дата звернення: 28.03.2025).

54) How to Use Roles and Manage Grant Permissions in PostgreSQL [Електронний ресурс]. DigitalOcean. 2022. Режим доступу: https://www.digitalocean.com/community/tutorials/how-to-use-roles-and-manage-grant-permissions-in-postgresql-on-a-vps-2 (дата звернення: 16.04.2025).

55) Using SSL / PostgreSQL JDBC Driver [Електронний ресурс]. Режим доступу: https://jdbc.postgresql.org/documentation/ssl/ (дата звернення: 21.04.2025).