МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЛІСОВИК ЄВГЕНІЙ СЕРГІЙОВИЧ

Допускається до захисту:
завідувач кафедри інформаційних
технологій
д.т.н., доцент
_____Т.В. Нескородєва
«__»_____20___р.

ЗАХИСТ ДАНИХ ЗА ДОПОМОГОЮ ПІДХОДУ ARMET

Спеціальність 105 Прикладна фізика та наноматеріали

Кваліфікаційна (магістерська) робота

(відповідно до стандарту спеціальності та ОП)

Науковий керівник:
В.Г. Крижановський, професор кафедри
Прикладної фізики та нанонематеріалів
д.т.н., професор
_____
(підпис)

Оцінка ____/_____/_____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК:_____
(підпис)

Вінниця 2022

АНОТАЦІЯ

Лісовик Є.С. Захист даних за допомогою підходу ARMET. Спеціальність 105 «Прикладна фізика та наноматеріали», Освітня програма «Технології інтернету речей». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній роботі досліджено методи та системи, проведено порівняльний аналіз двох спільних систем та виявлення певних переваг та недоліків, на основі яких розроблено додаток для керування, регулювання та відновлення системи у побутовому домашньому кондиціонері. Підхід Armet дає змогу контролювати коректну роботу системи, та в разі хакерської атаки, мати змогу відновлення роботоспроможності системи.

Ключові слова: захист даних, автоматизована система, хакерська атака, PID, RFID, RSM, Java, MySQL, MySQL Workbench.

51 с., 13 рис., 31 джерело.

SUMMARY

Lisovyk Y. S. Data protection using the ARMET approach. Specialty 105 "Applied Physics and Nanomaterials", Educational Program "Internet of Things Technologies". Vasyl' Stus Donetsk National University, Vinnytsia, 2022.

The qualification work investigates the methods and systems, conducts a comparative analysis of two common systems and identifies certain advantages and disadvantages, on the basis of which an application for controlling, regulating and restoring the system in a domestic home air conditioner was developed. The Armet approach allows you to control the correct operation of the system, and in the event of a hacker attack, to be able to restore the system's performance.

Key words: data protection, automated system, hacker attack, PID, RFID, RSM, Java, MySQL, MySQL Workbench.

Pages 51, Fig. 13, Bibliography: 31 items.

ЗМІСТ

ВСТУП

IoT – це усе, що надає надзвичайний зв'язок між різними об'єктами в найрізноманітніших галузях. Як уже говорилось раніше, низка перспективних та надійних тенденцій в інформаційному просторі заклали потужну і міцну основу для візуалізації перспективних ідей в Інтернеті Речей.

Одна з найголовніших проблем є забезпечення надійності мережі, тому що інформація передається ще й по бездротовій мережі, котра є менш надійною в наслідок використання відкритого простору. В даний момент це є гостра проблема та елементи незаконного бізнесу. Об’єктом дослідження є Інтернет речей (IoT). Предмет дослідження - методика (алгоритм) для забезпечення максимально надійного захисту розумного дому.

Проблема, що вирішується – забезпечення інформаційної безпеки розумного дому, захист даних в різних умовах, впливу навколишнього середовища, а також фізичного втручання в систему.

Актуальність – у зв'язку з появою всесвітніх поширюваних комп'ютерних мереж і створенням великого віртуального всесвіту, знайшлась можливість викрадення інформаційних даних та ресурсів у незахищених користувачів. В цьому випадку ключове питання є забезпечення не лише безпеки, а й надійної роботи пристроїв в цих системах.

Мета – створення програмного забезпечення для захисту даних від кібератак та хакерів [1] і можливість відновлення втрачених матеріалів за допомогою ARMET підходу для безпеки окремих приладів системи “Розумний дім” [2].

Об'єкт – виявлення атак та забезпечення відновлення даних.

Завдання – зробити аналітичний огляд літератури, дослідити підхід ARMET, розробити програмне забезпечення з використанням мови програмування Java, створити алгоритм виявлення атак інструментами технології RSM.

Наукова новизна – створення алгоритму для виявлення кібератак інструментами технології RSM, адаптивне резервне копіювання даних, розробка програмного забезпечення для керуванням даними.

РОЗДІЛ 1 СУЧАСНИЙ СТАН ЗАБЕЗПЕЧЕННЯ ЦІЛІСНОСТІ ДАНИХ В IoT

1.1 Призначення технології IoT

Однією із головних концепцій Інтернету речей – це допомага жити і працювати доступніше, і також одержувати кращий нагляд за своїм життям. На додаток до розумних пристроїв для автоматизації будинків, IoT є важливим для бізнесу [3]. IoT надає компаніям можливість у реальному часі вивчити, як реально працюють їх системи, котрі можуть надавати візуальні можливості про велику кількість інформації – від продуктивності машин до ланцюгів поставок та логістичних операцій. IoT дозволяє великим компаніям автоматизувати роботу та певні процеси та зменшити споживання ресурсів та кількість робочої силу. Це дає змогу скоротити витрати відходів та покращує надання послуг, роблячи дешевшим виробництво та доставку товарів, а також забезпечуючи прозорість операцій із клієнтами. Таким чином, Інтернет-речей є однією з найнеобхідніших технологій як у світі так і в повсякденному житті, і він буде продовжувати вдосконалюватись та оптимізовуватись, оскільки більша кількість підприємств розуміє великий потенціал приєднаних до мережі пристроїв, щоб зберегти їх конкурентоспроможність.

Перелік непередбачуваних викликів перешкоджає надійності роботи пристроїв IoT та забезпеченню надійної безпеки в середині середовища IoT [4]. Оскільки основною ідеєю мережевих приладів та інших об'єктів є відносно новою, безпечність таких систем не завжди вважалася першочерговим завданням на етапі проектування продукту. Оскільки IoT є відносно новим ринком, багато спеціалістів та виробників продукції зацікавлені як найшвидше почати випускати свою продукцію на ринок, а не втрачати час вдосконалюючи система на захищеність. Основною проблемою безпеки таких пристроїв, що вони часто обмежені певними ресурсами або ж взагалі не містять обчислювальних ресурсів, які необхідні для забезпечення надійності безпеки. В результаті чого, багато пристроїв зовсім не мають або не можуть запропонувати розширені функції

безпеки [5]. Один із прикладів це - датчики, які можуть контролюють вологість або температуру повітря. Вони не мають можливості подолати розширене шифрування або іншими заходами безпеки. Плюс до того, що багато пристроїв IoT розміщені в не до кінця продуманих місцях до кінця життя, вони навряд чи зможуть отримувати актуальні оновлення безпеки. З точки зору виробника, створення безпеки з самого початку може бути занадто дорогим, зменшити швидкість розробки і виробництва, і як насклідок призвести до того, що пристрій не працюватиме як слід.

«Розумний будинок» - це практично незалежне автоматизоване житло, що включає в себе велику кількість вбудованих технологій [6]. Технології «розумного дому» можуть контролювати внутрішнє середовище та дії, які застосовуються коли на це є необхідність. Основним результатом цих модифікацій є те, що «розумний будинок» має здатність відслідковувати дії мешканців середовища в якому він працює, незалежно керувати пристроями у ньому за на перід вказаними правилами, які визначені користувачем [7]. Це може бути як цільна система усього дому, а також робота однією із приладів, наприклад кондиціонер з віддаленим доступом.

В великій кількості випадків технологія розумного будинку є достатньо гнучкою, щоб мати здатність інтегруватись із достатньо великим спектром приладів та правил. Базова архітектура дозволяє отримувати найрізноманітніші показники в домашніх умовах, оброблювати дані, використовуючи датчики з мікроконтролероми[8] для виміру людських показників життєдіяльності у будинку, за допомогою механізмів для моніторингу домашніх пристроїв. Найбільш ідеальна архітектура використовується як певний шаблон для розробки конкретної архітектурної топології даного середовища, оскільки вона дає змогу забезпечувати загальну базу, на основі якої можуть бути розроблені бази більш складної архітектурної системи [9].

1.2 Підхід Armet

Перспективний підхід, який об’єднує безпеку та безпеку в Інтернеті речей, системи та кібер-фізичних систем є підходом ARMET [10]. В основі АРМЕТ базується три основних концепції: створення безпечних систем, контроль даних систем для правильної роботи та виявлення атак або збоїв, і у разі збою або атаки ми можемо мати плани відновлення, залежно від проблеми та кількості інформації про неї. АРМЕТ був розроблений з акцентом на промислові системи управління [11], але вона також застосовується для системи Інтернету речей, оскільки їхня програмна складність порівнянна зі складністю програмного забезпечення промислової системи управління.

Завдяки підходу ARMET, додаток IoT [12] розробляється, починаючи з виконуваної специфікації, яка доказово узгоджується з набором властивостей безпеки для застосування. З цієї специфікації виконуваного файлу код є похідним програми. Враховуючи специфікацію виконуваної програми та код програми для цільової системи, ARMET відстежує поведінку програми під час її виконання, порівнюючи її спостережувану поведінку з очікуваною поведінкою на основі специфікацій програми [13]; щоб досягти цього, проміжне програмне забезпечення виконує виконуваний файл специфікації паралельно з виконанням програми в системі IoT і обчислює прогнози поведінки програми.

Вона складається з кількох компонентів: монітору безпеки [14] під час виконання, модуль діагностики, модуль відновлення, модель довіри, модуль вибору адаптивного методу та модуль резервного копіювання. Монітор безпеки під час виконання є важливим компонентом проміжного програмного забезпечення, яке приймає як вхідні дані виконуваної специфікації програми та стан системи, яка виконує код програми. Монітор спостерігає за поведінкою виконання програми і, паралельно, прогнозує стан програми виконання шляхом виконання його специфікації; виконання специфікації визначає очікувана «хороша поведінка» програми і, за бажанням, відома «погана поведінка» програми, яка містить відомі атаки. Порівнюючи передбачення із спостереженням, монітор може виявити відхилення, які вказують на збій

програми або атаку. Коли таке виявлення зроблено, ARMET переходить до етапу діагностики, щоб визначити збій або атаку на основі моделі довіри, що воно включає. Після завершення етапу діагностики вся доступна інформація використовується модулем відновлення. На основі діагностичної інформації одужання модуль вибирає найоптимальніший адаптивний метод відновлення і дає можливість системі відновлюватися, враховуючи попередні стани, збережені в резервній копії модуля. Важливо відзначити, що система буде працювати за будь-яких сценаріїв невдачі та атаки, навіть невідомі, тобто при невдачах та атаках, які не було очікувані та не включені в модель довіри. У гіршому випадку, коли модуль діагностики не надає корисної інформації, система відновиться шляхом повернення до попереднього чистого стану. Крім того, підхід базується на одному припущенні: специфікація виконуваної програми виконується в безпечному середовищі і не може бути атакована, тобто її передбачення завжди вірні.

Але, незважаючи, на всі переваги даного підходу, час не стоїть на місці і кожного дня створюються нові методи отримання захищених даних, тому і потрібно покращувати їх якість. Це може бути додаткові фактори захисту, вдосконалена система від збоїв та атак або додаткове шифрування даних [15].

1.3 Порівняльна характеристика з IoT в медицині

Пристрої Інтернету речей (IoT) здебільшого можна використовувати для полегшення віддаленого моніторингу здоров’я та системи екстреної медичної допомоги. В даний час ми стикаємося з багатьма проблемами в реальному світі, які вимушені поводитися реально.

За допомогою IoT відновлюються проблеми, що забирає більше часу, ресурсів і робоча сила. В останні роки набуває популярності, а також розвиток Інтернет-ресурсів розумних програм, таких як розумний дім. У порівнянні з традиційною системою, розумна реабілітація є націленою при забезпеченні

ефективного лікування, достатньої взаємодії та швидкої реконфігурації, щоб визначитися можливе використання медичних ресурсів відповідно до конкретних потреб пацієнта. Інтернет речей – це первинна технологія взаємозв'язку всіх медичних ресурсів систем реабілітації.

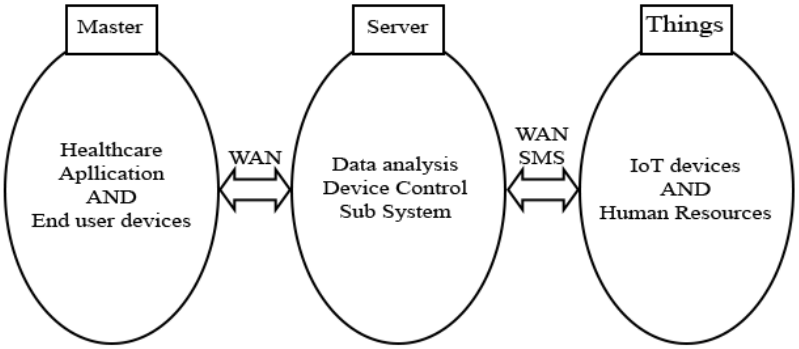


Рис. 1.1 Інтернет речей в медицині

Також поєднувати мережеві технології, які дають змогу широкому спектру додатків, пристроїв чи речей взаємодіяти та спілкуються між собою. Інтернет речей (IoT) має багато форм застосування, включаючи системи охорони здоров’я, як показано на (Рис. 1.1) та промислові системи. Так, системи охорони здоров’я, які в основному використовують взаємопов’язані пристрої для створення мережі IoT, яка захоплюється оцінкою, автоматично виявляє ситуацію та контролює пацієнтів, де медичні втручання є обов'язковими. Отже, лише IoT може сформувати інформаційну мережу, яка з’єднує лікарні, людей, медичні прилади, будинок оточення та інші робочі місця.

Впровадження Інтернету речей є гнучким і доступним результатом, що дозволяє охороні здоров'я програми для обслуговування пацієнтів з кращим лікуванням, також зроблені за допомогою дистанційного моніторингу пацієнтів та ефективні обробка медичних даних. Для реалізації медичної допомоги необхідні деякі різні характеристики обслуговування в навколишньому середовищі.

1) Оцінка онтології:

У цій частині вхідні дані перетворюються на функцію, в якій аналізуються симптоми пацієнтів, захворювання розташована, і вся інформація поміщається у віддалену базу даних. Коли пацієнт вперше потрапляє до лікарні, то фізичні характеристики захворювань впершу чергу буде визначати лікар. Визначені функції поділяються на класи та підкласи. Заняття представляють пацієнтів основні інформація та підкласи представляють детальну інформацію про захворювання.

2) Глобальне порівняння онтології:

У цій оцінюваній онтології порівняно з глобально збереженою онтологією захворювань на основі система бази знань. Глобальна онтологія містить дві форми онтології, це онтологія хвороби та онтологія ресурсів. Онтологія захворювання містить основну та медичну інформацію та ресурсну онтологію пацієнта охоплює такі медичні ресурси, як лікарі, медичні прилади тощо.

3) Розрахунки подібності:

Введіть зміст запиту про симптом вручну. Порівняйте вхідний симптом з усіма глобальними онтологія захворювання в базі знань. Виконайте автоматичне визначення схожості симптомів для відповідність подібності. Автоматизуйте вибір найбільш схожого випадку в базі знань. Це легко дізнатися які типи пристроїв необхідні для відповідних симптомів та стратегій реабілітації через онтологія.

4) Оптимізація дизайну:

Завершальна фаза процесу розумного проектування, на якій визначаються процедура, методи, обсяги та тривалість окремі дії. Оптимізація параметрів має важливе значення, щоб переконатися, що деталі зустрічаються вимоги до структури. Автоматичні конструкції допомагають побудувати систему, де новий пацієнт міг би швидко опинитися діагностовано, відповідна стратегія допомоги незабаром може бути розроблена, а також пов’язані медичні властивості

розповсюджується в короткий термін. З онтологією, що забезпечує добре впорядковану структуру знань.

Для реалізації запропонованої системи необхідно мати деякі різні форми розрізи, показані на (Рис.1.2.) Перший – це взаємодія людини з машиною, а другий – мультидисциплінарній оптимізації, яка формується в багатьох операціях над архітектурою системи. І третій – управління додатків базою даних і відображення класів у базі знань. Взаємодія людини з машиною може бути досягнуто базою ресурсів і люди, як лікарі, медсестри та пацієнти, є реальними людьми ресурси та пристрої, такі як RFID[16], швидка допомога, медичні ресурси, взаємодіють з людиною ресурси. По-друге, багатодисциплінарна оптимізація, яка використовується для виконання проектування автоматизованого проектування методології і головну роль в архітектурі системи, оскільки вона створює всю стратегію системи і також для автоматичного надання рецепта пацієнту. По-третє, керування програмами використовується для керування всіма ресурси та записи пацієнтів також. Облік пацієнтів також ведеться за класами та підкласами як пояснюється при реалізації.

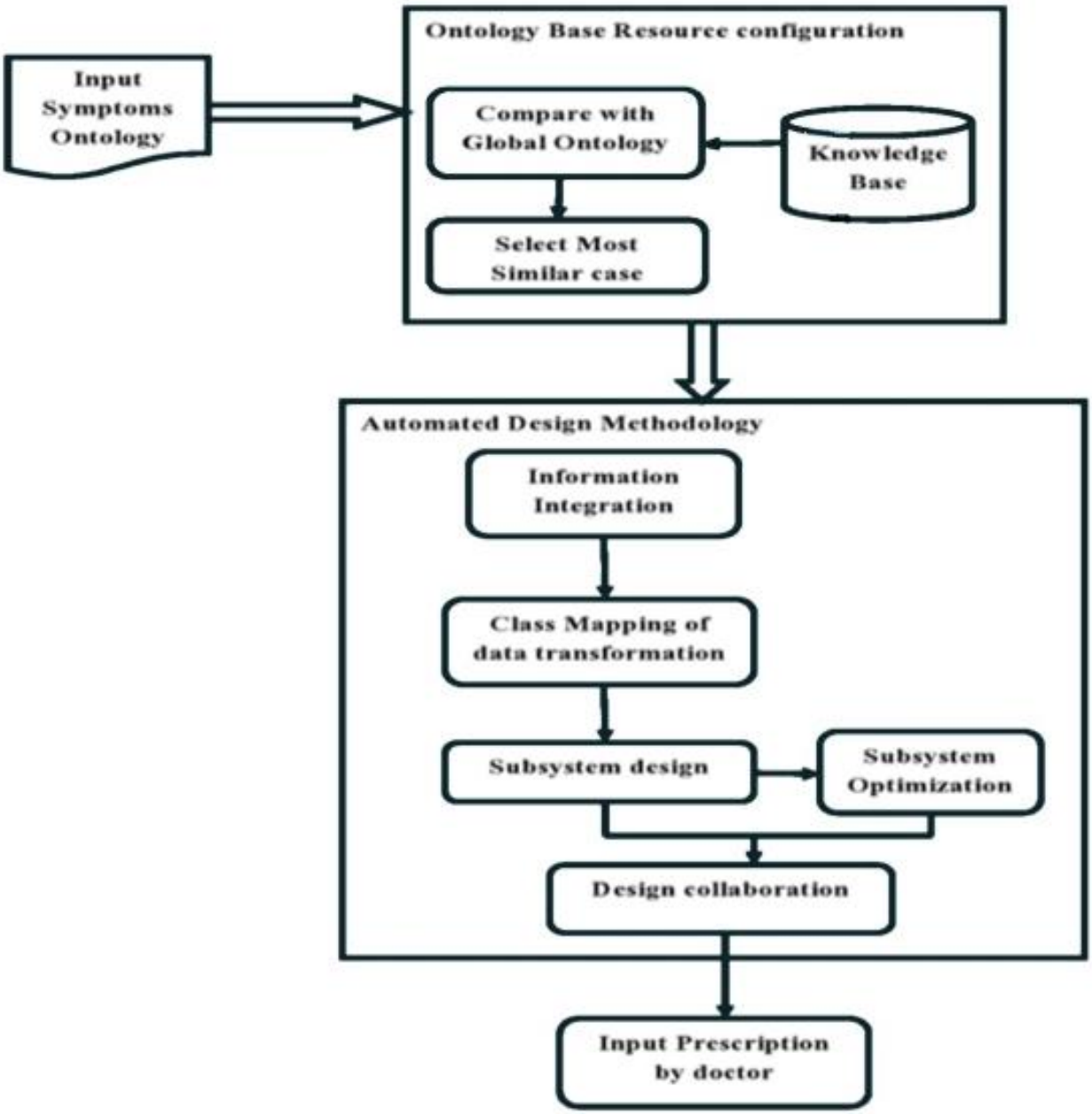


Рис. 1.2 Системна архітектура

У управлінні додатками також виконується конструкторська робота, інформація і інтеграція додатків на основі бази даних і бази знань у системі.

Генерація методу автоматизованого проектування повинна мати деякі етапи, пояснені в виконання та супутні роботи. Коли пацієнт або лікарі вводять симптоми захворювання, це може бути порівняти з глобальною онтологією на основі бази знань і вибрати їх найбільш схожий випадок у знаннях бази. Після автоматизованого проектування методологія відображення на класи та підкласи вже збережених пацієнтів записи.

А підсистема може оптимізувати записи пацієнтів на основі їхніх захворювань та основної інформації. Нарешті конструкторська співпраця може відігравати важливу роль у схемі дизайну (рис 1.3) для надання рецепту пацієнту після перевірка лікарів. Використання цієї системи може бути ефективним для медичної системи та систем охорони здоров'я.

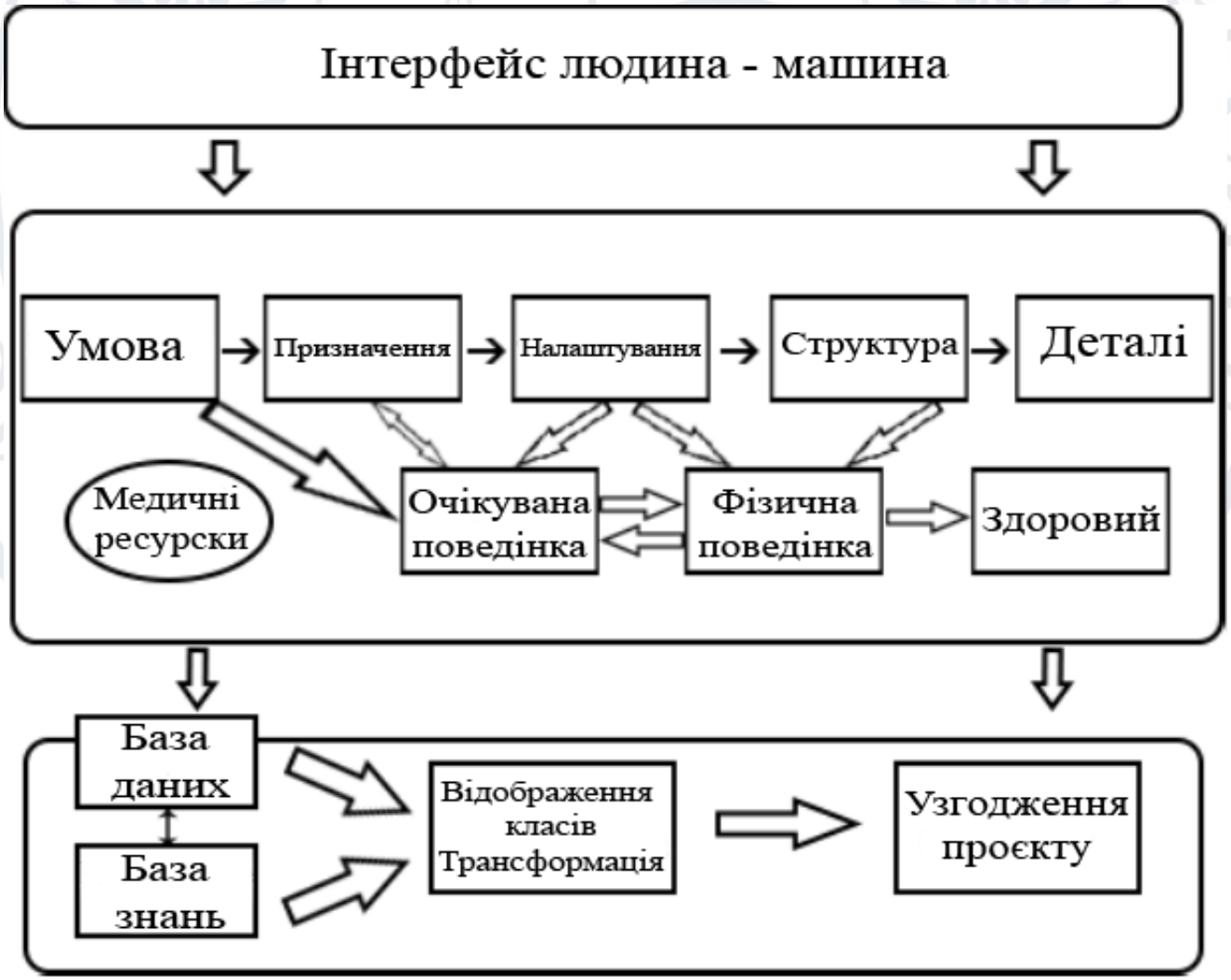


Рис. 1.3 Процес генерування методу автоматизованого проектування

1.4 Висновки за розділом 1

У цьому розділі розглянуто етапи актуального стану задач, моделей та різноманітних методів вирішення. На основі чого, була поставлена задача, основою якої була вимога розв'язати проблеми впровадження розумних систем та виявлення не коректної роботи у приладах, що не мають великої потужності. Через надвелику кількість приладів та недостатньої кількості даних про їх несправність, вирішено розробляти системи на основі їх коректної роботи.

А також було порівняно схожі між собою моделі, на основі яких можна зробити наступні висновки:

1. Модель Armet у своїй методології має кластер захисту даних, який має цикл обробки даних на наявність помилкових або некоректних даних.

2. Обидві системи мають бази даних, але медична система поступається системі Armet, тим що наша система має додатковий цикл оновлення бази даних.

3. Система Armet має адаптивний вибір вводу даних та пошуку і обрання методу.

4. Методології Armet також має розширений функціонал моделі довіри.

5. Медична система, в свою чергу, має більшу базу даних, що дає можливість у нечастому оновленні інформації.

РОЗДІЛ 2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Концептуальна модель

Розглянемо модель на прикладі побутового кондиціонеру Samsung з віддаленим доступом роботи. На основі якого буде будуватись алгоритм роботи [17].

Монітор безпеки під час виконання програмного забезпечення, яке приймає як вхідні дані виконуваної специфікації програми та стан системи, яка виконує код програми. Монітор спостерігає за поведінкою виконання програми і, паралельно, прогнозує стан програми виконання шляхом його специфікації; виконання специфікації визначає очікувана «хороша поведінка» програми і, за бажанням, відома «погана поведінка» програми, яка містить відомі атаки. Порівнюючи передбачення із спостереженням, монітор може виявити відхилення, які вказують на збій програми або атаку.

Коли таке виявлення зроблено, система переходить до етапу діагностики, щоб визначити збій або атаку на основі моделі довіри, що воно включає. Після завершення етапу діагностики вся доступна інформація використовується модулем відновлення.

На основі діагностичної інформації одужання модуль вибирає найоптимальніший адаптивний метод відновлення і дає можливість системі відновлюватися, враховуючи попередні стани, збережені в резервній копії модуля.

Важливо відзначити, що система буде працювати за будь-яких сценаріїв невдачі та атаки, навіть невідомі, тобто при невдачах та атаках, які не було очікувані та не включені в модель довіри.

У гіршому випадку, коли модуль діагностики не надає корисної інформації, система відновиться шляхом повернення до попереднього чистого стану. Крім того, підхід базується на одному припущенні: специфікація виконуваної програми виконується в безпечному середовищі і не може бути атакована, тобто її передбачення завжди вірні(Рис. 2.1).

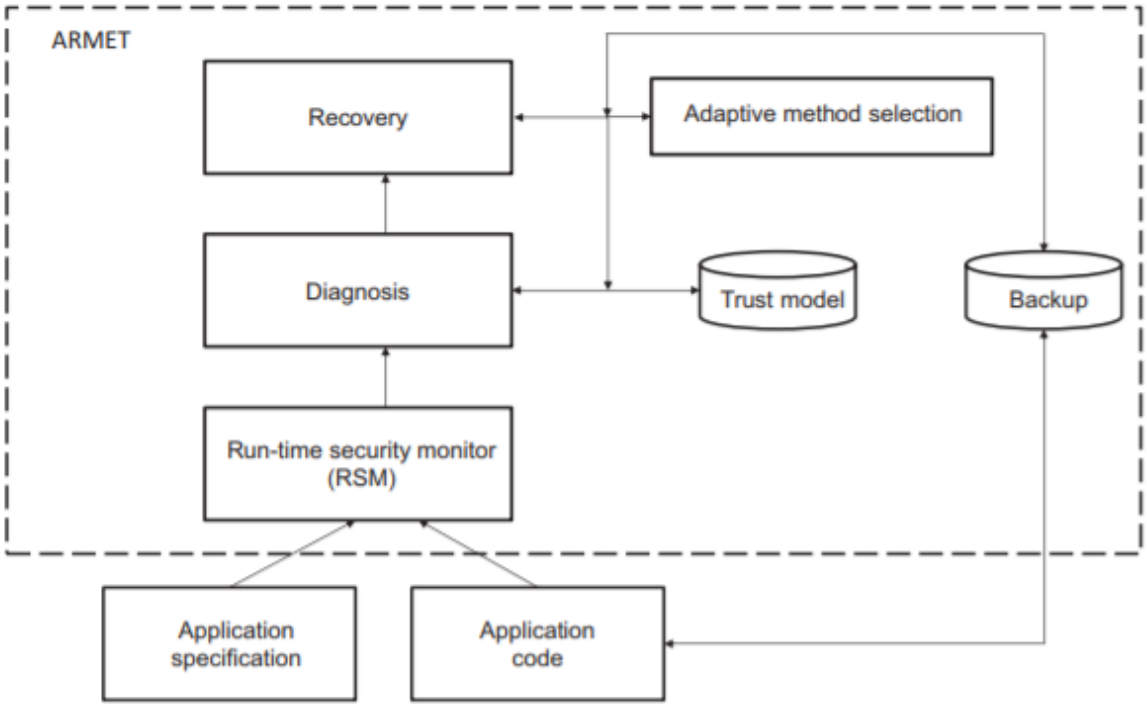


Рисунок 2.1 – Схема роботи алгоритму

Дерево рішень будується за допомогою чотирьох етапів:

1) Спостереження за поведінкою програми за допомогою RSM.
2) Визначення збою або атаки за допомогою моделі довіри.
3) Вибір найоптимальнішого адаптивного методу відновлення.
4) Повернення до попереднього чистого стану виконання роботи (FullBackup).

2.2 Логічна модель додатка

Для написання даного програмного забезпечення була обрана мова програмування Java [18]. Це мова спеціально створювалась для розробки додатків і володіє усіма технологіями для реалізації динамічного обміну даними між різними системами розподілених мереж.

Java є відносно простою мовою програмування. Котра дає можливість швидко створювати надійний програмний код, який буде сприяти автоматичному керуванню пам'яттю.

Байтовий код Java в більшій мірі немає залежності від операційниї систем або процесорів. Завдяки цьому додатки на мові Java [19] легко портуються на різні платформи.

Для зберігання даних використовується база даних MySQL[20]. MySQL - компактний багатопотоковий сервер баз даних. Характеризується високою швидкістю, стійкістю і простотою використання. MySQL вважається гарним рішенням для малих і середніх застосувань. Перевагами цієї бази даних є

• легке встановлення та використання;

• можливість підтримувати велику кількість користувачів, що одночасно використовують бази даних;

• наявність рядків в таблиці може сягати декількох мільйонів;

• шкидкісне виконання команд;

• можливості ефективної і надійної системи безпеки.

Після того, як ми нормалізуєм базу даних, схема буде мати наступний вигляд (Рис.2.3). Щоб адаптивно використовувати програму, юзер під час перегляду даних отримує інформацію, яка поєднуються складними SQL-запитами.

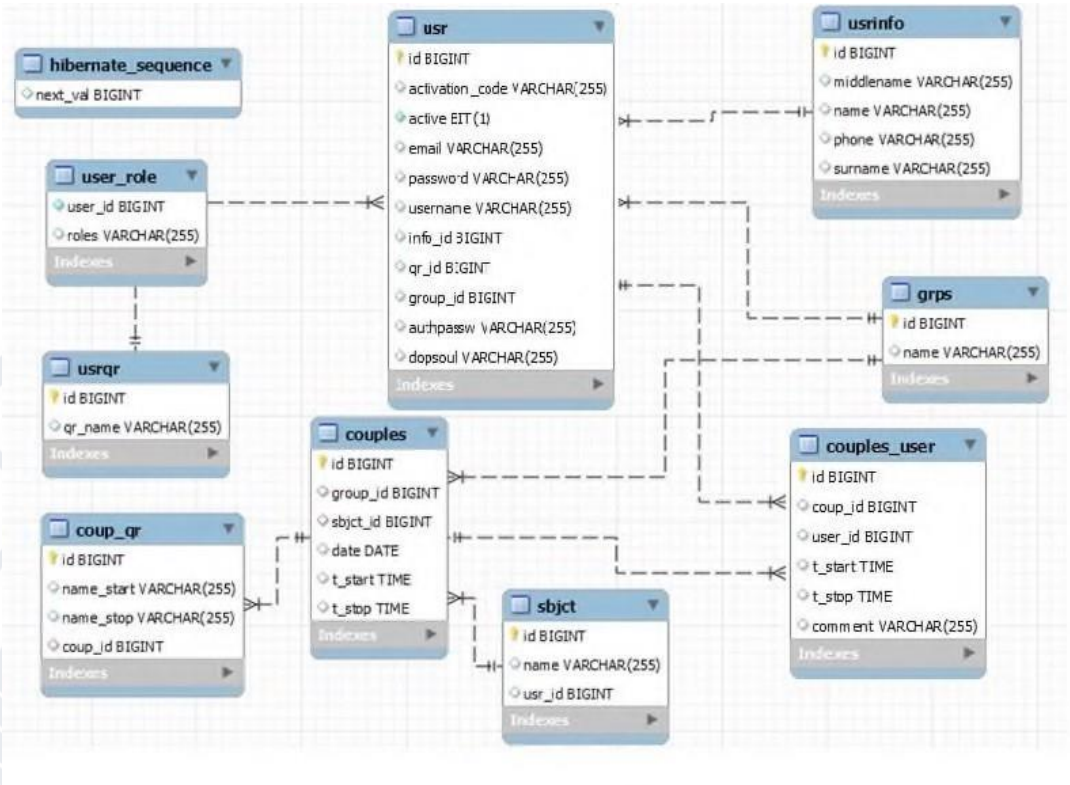


Рисунок 2.3 - Схема бази даних після нормалізації

Обмеження, пов’язані з використанням мережевих технологій:

- Кожні підключення до БД з ПЗ повинні бути унікальним, саме тому, для юзера даного забезпечення створюється новий користувач БД з своїми обмеженнями з точки зору БД.
- Необхідно знати IP адресу до сервера БД, щоб мати змогу підключитись.
- Необхідність знаходитись в одній мережі додатку з БД.

Діаграма пакетів зображена на наступному рисунку (Рис.2.4). Діаграма класів для функціоналу показана на рисунку (Рис.2.5) .

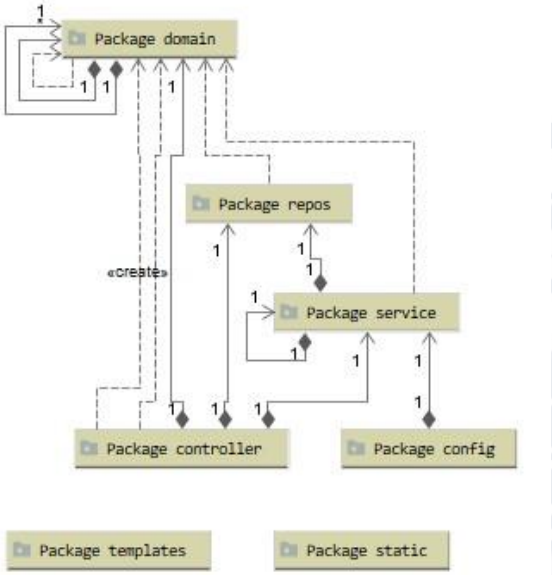


Рисунок 2.4 - Діаграма пакетів

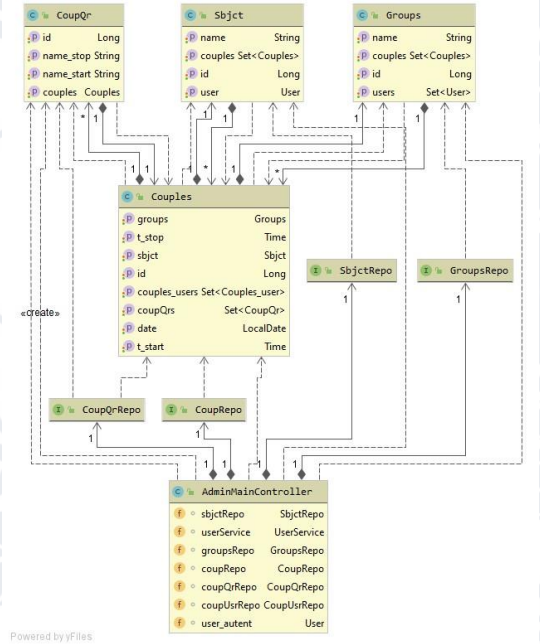


Рисунок 2.5 - Діаграма класів для функціоналу.

2.3 Фізична модель додатка

Щоб зберігати дані було обрано СУБД MySQL. MySQL - це реляційна система управлінням базами даних. Дані в цих базах зберігаються у вигляді логічно-пов'язаних між собою таблиць, доступність до котрих виконується за допомогою мови запитів SQL. MySQL - легко поширювана система, вносити

платню за її використання не потрібно. Адже, це достатньо швидка, надійна і проста у використанні СУБД, вона найкраще підходить для невеликих проєктів.

Щоб налаштувати базу даних із серверної частини, використаємо MySQL Workbench - інструмент для візуального проектування баз даних, що інтегрує проектування, моделювання, створення й експлуатацію БД в єдине безкоштовне оточення для системи баз даних MySQL[21].

За реалізацію клієнт-серверної взаємодії відповідає драйвер JDBC[22] та файл properties в додатку, в якому описується підключення та взаємодія з сервером БД.

Використаємо шаблоні налаштування до сервера БД. Найнеобхідніше налаштувати параметр «wait_timeout=7200» чи більше, це залежить від часу роботи даної програми, і щоб уникнути проблем розриву стабільного з'єднання з сервером. Можна установити параметр «default-stoarage-engines=InnoDB»[23] бо він є більш потрібнішим до великих об'ємів даних.

Коли створюємо новий запит БД необхідно налаштувати права користувача, а саме доступ тільки до запитів Delete, Update, Select.

Підключення БД до програми:

```
jpa.hibernate.dll-autoupdate
    datasource.url=jdbc:mysql://
${MYSQL_HOSTS:XXX.XXX.XXX.XXX}:3502/room?autoReconnect=true&u
seSSL=false&allowPublicKeyRetrieval=true
 datasource.username=******
datasource.password=******
datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

2.4 PID-регулятор

В ході роботи вагомою потребою стало запровадження контролера, який би контролював затрату часу. Частиною підходу ARMET є PID – регулятор, який був впроваджений як окрема, незалежна складова RSM.

У реальному робочому середовищі PID буде працювати як додаток у програмованому логічному контролері, а наш RSM працюватиме як частина проміжного програмне забезпечення. Таким чином, програма PID буде розроблена через будь-яке середовище розробки додатків, наприклад, з використанням драбинної логіки, тоді як RSM буде частиною операційної системи.

Був проведений експеримент, який проходив на MacBook Pro з процесором Intel Core i7 з частотою 2,8 ГГц.

Припустимо, що PID - контролер кожен раз запускає цикл керування в підсистемі довжиною 0,1 с, що є дуже низькою частотою циклів керування для такого контролера. Наш експеримент моделює підсистему протягом 100 с, тобто 1000 циклів керування, і ми спостерігаємо це, коли вмикаємо детальний моніторинг, RSM споживає 8.93×10^{-4} с часу процесора і 9.07×10^{-4} с реального часу для кожного циклу алгоритм PID, який дуже малий (менше 2%) порівняно з тривалістю циклу керування (0,1 с).

Для демонстрації, ми змоделювали PID-регулятор із чотирма параметрами: задане значення та три вагові коефіцієнти Kp, Ki і Kd.

Виконаємо наступні дії:

1) Використаєм значення датчика, щоб оцінити стан системи.

2) Обчислити різницю між оцінюваною системою

стану і задане значення контролера (термін помилки).

3) Обчисліть локальну похідну похибки.

4) Інтеграція помилки.

5) Обчислити відповідну поправку:

а) похибку помножити на Kp;

б) інтеграл похибки помножити на Ki;

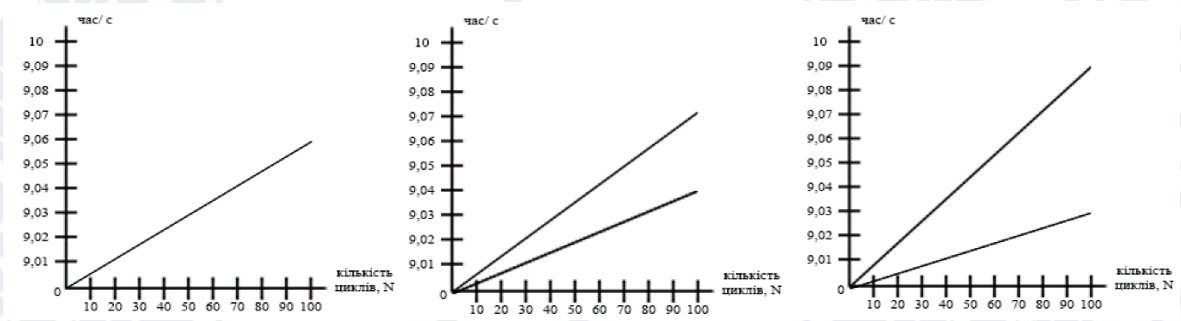
в) похідну похибки помножити на Kd.

6) Обчисліть і виведіть суму цих трьох умов як термін корекції.

Per 1×10^{-1} Cycle	CPU	Real-time
Full RSM	8.93×10^{-4}	9.07×10^{-4}

Рисунок 2.6 – Продуктивність виконання роботи

На основі даного експерименту, виходячи з даних, можна графічно зобразити залежність виконання циклів:



А) Б) В)

Рисунок 2.7 – «Контроль втручання в роботу циклу»: А) цикл відпрацював коректно; Б) максимальне допустиме відхилення; В) стороннє втручання в роботу циклу

2.5 Висновки за розділом 2

В даному розділі було сформовано головні сутності для коректної роботи з базами даних, на основі яких вона була створена і складається з нормалізованих таблиць до 3 Нормальної Форми.

Виявлено основні ролі на основі яких створено сценарій роботи додатку, паралельно розроблялась база даних та програмний код, котрий має структуровані пакети та патерни.

Наша база даних була автоматично підключена до програмного забезпечення, були створені доступи до мережі з унікальними рівнями доступу до БД.

Проведений експеримент для визначення ефективності PID-регулятора[24] в комплексі з RSM[25], а також проведений аналіз коректної роботи, максимального відхилення, та недопустимого відхилення.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Опис засобів реалізації

Для реалізації рішення можна виділити основні етапи:

1. Реалізація бази даних та їхня наповненість.
2. Реалізація програмного додатка.
3. Підключення бази даних.
4. Налаштування режиму доступності.
5. Авторизація та аутентифікація на усіх необхідних рівнях.
6. Налаштування авторизації та аутентифікації через зовнішній пристрій.
7. Аналіз розробленого додатка.
8. Модернізація додатка на основі проведеного аналізу.

3.2 Налаштування авторизації системи та користувацького режиму

Для налаштування користувацького режиму на сервері бази даних були створені унікальні користувачі з обмеженим доступом. Користувач, котрий підключається, може давати тільки декілька типів запиту Select, Update, інші – неможливо. Також було налаштовано IP – адрес, з котрого можна підключатися до БД. Таким чином ми можемо створювати рівень надійності з точки зору доступу до бази даних ПЗ.

Для авторизації та аутентифікації користувача у системі використовується Security.

```
<depandency>
  <groupId>org.framework.boot</groupId>
  <artifactId>boot-starter-security</artifactId>
</dependency>
```

Основний об'єкт - SecurityContextHolder. В цьому об’єкті зберігаються дані про актуальний пакет безпеки забезпечення, котрий вміщує в себе потрібні пакети даних про юзера, який працює з додатком. Security застосовує об'єкт Authentications, користувач авторизованої сесії.

Аутентифікація: юзеру надасться можливість зайти до системи надав login та password. Вони з’єднуються у екземпляр класу UsernamePasswordAuthenticationToken(екземпляр інтерфейсу Authentication) після чого він передається екземпляру AuthenticationManager для перевірки.

Щоб створити UserDetails потрібно використати інтерфейс UserDetailsService, з одним методом:

@Override public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {

return userRepro.findByUsername(username);

}

3.3 Налаштування циклу системи

В даному блоці створюється булева зміна «isAtack» , яка має два стани: хороша поведінка – true і погана поведінка – false (яка стоїть по замовчуванню). Порівнюючи передбачення із спостереженням, монітор може виявити відхилення, які вказують на збій програми або атаку. В даному випадку блок звертається до об'єкту bug, який прописаний в окремому класі, і має такі параметри, як ім'я, естімейт, історію і повертає баг з заповненими параметрами.

```
Pair applicationStory = null;
Pair userStory1 = applicationStory; {
   if (applicationStory == null) {
      return null;
   } else {
```

```
        Object id = null;
        Bug bug = new Bug(id, armetName, estimate, applicationStory);
        bug.name = armetName;
        bug.estiate = estimate;
        bug.userStoryId = applicationStory.id;

        return bug;
    }
}
```

Другий блок - після завершення етапу діагностики вся доступна інформація записується в файл. Також в тому блоці присутні виключення. Необхідно для того, щоб програма працювала, в будь-якому існуючому з двух кейсів: невдачі та атаки, або нормальна поведінка програми.

```
if (codeArmet == fullProgram) {
    int i;
    for (i = 0; i < codeIndex; i++) {
        char c = armetBlock(i);
        FileReader fr = fr1;
        fr = fr1;
        try {
            File file = new File("Armet.java");
            fr = new FileReader(file);
            char[] a = new char[lngth];
            fr.read(a);
            for (char h: a) {
```

```
            System.out.print(c);
        }
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        try {
            fr.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}
```

Третій блок - відбувається зчитування даних з попереднього блоку, аналіз на основі діагностичної інформації за допомогою якої програма вибирає найоптимальніший адаптивний метод відновлення з виключеннями які також дозволяють працювати програмі в різних кейсах.

```
FileReader fr = fr1;
    try {
        File file = new File("Armet.java");
        fr = new FileReader(file);
        char[] a = new char[lngth];
        fr.read(a);
        for (char c: a)
            System.out.print(c);
    } catch (IOException e) {
        e.printStackTrace();
```

```
    } finally {
        try {
            fr.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
```

В четвертому блоці вирішується, чи модуль діагностики не надає корисної інформації, система відновиться шляхом повернення до попереднього чистого стану

```
if (codeArmet == true) {
        int nextId = 1;
        Clear clear = new Clear();

        assertTrue(clear.returnToPrevious(new armetStory(nextId++, count)));

        Content[] content = content.getArmet();

        assertEquals(j, content.length);
        assertEquals(content[j].toString());
        assertEquals(j, Clear.getContent());
    }
} else {
    armet.isClear
}
}
```

3.4 Візуальне макетування проєкту

Для розробки проєкту використовувався фреймворк SceneBuilder.

SceneBuilder інструмент який дозволяє швидко проєктувати користувацькі інтерфейси додатків JavaFX без кодування.

Користувачі мають змогу перетягувати різні елементи інтерфейсу користувача до робочої області, змінюючи їх властивості, використовувати таблицю стилів, а також код FXML для макетування, котрий вони створюють, автоматично створюється у фоновому режимі. Результатом якого є файл FXML, котрий можна буде об'єднати з проєктом Java, прив'язати інтерфейс користувача до логіки програми [26].

Зробити адаптивним графічний інтерфейс дуже важливо. Сьогодні розміри пристрою варіюються від портативних до настінних телевізорів. Використовуючи правильні контейнери та конфігурації компонентів, можна швидко розробити графічний інтерфейс JavaFX за допомогою конструктора сцен.

За допомогою таблиці стилів можна застосувати свій вигляд до макета графічного інтерфейсу. Це так само просто, як вибрати компонент графічного інтерфейсу та вказати на файл CSS в панелі властивостей. Аналізатор CSS дозволяє зрозуміти, як конкретні правила CSS можуть впливати на аспекти компонента JavaFX[27].

З JavaFX, немає необхідності вивчати додаткові технології, оскільки попереднє знання будь-якої технологій буде достатньо хорошим для розробки RIA[28] з використанням JavaFX.

Код програми JavaFX може посилатися на API з будь -якої бібліотеки Java. Наприклад, програми можуть використовувати бібліотеки API[29] Java для доступу до власних системних можливостей і для підключення до програм на основі проміжного програмного забезпечення на основі сервера.

Зовнішній вигляд додатків JavaFX можна налаштувати. Каскадні таблиці стилів (CSS) відокремлюють зовнішній вигляд і стиль від реалізації, щоб розробники могли зосередитися на кодуванні. Графічні дизайнери можуть легко налаштувати зовнішній вигляд та стиль програми за допомогою CSS[30]. Якщо

присутній фон веб-дизайну, або якщо необхідно відокремити інтерфейс користувача (UI)[31] і логіку сервера, то є можливість розробити презентаційні аспекти інтерфейсу на мові сценаріїв FXML та використовувати код Java для програмної логіки. Якщо надається перевага розробці інтерфейсів користувача без написання коду, використовується JavaFX Scene Builder. (рисунок 3.1).

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.image.Image?>
<?import javafx.scene.image.ImageView?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
            minWidth="-Infinity" prefHeight="399.0" prefWidth="709.0"
            style="-fx-background-color: #2E3348;" xmlns="http://javafx.com/javafx/16"
            xmlns:fx="http://javafx.com/fxml/1"
```

Рисунок 3.1 – Фрагмент коду, де відбувається об’єднання файлу FXML з Java

Створюючи новий проект, можна підключити JavaFX, що є вбудованою функцією IntelliJ IDEA. Основним виконувачем програми являється клас HelloAplication, в якому вмикається зв'язок з FXML файлом (рисунок 3.2).

```
package com.example.armet

import ...

public class HelloApplication extends Application {
    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource( name: "hello-view.fxml"));
        Scene scene = new Scene(fxmlLoader.load(), v: 700, v1: 400);
        stage.setTitle("ARMET");
        stage.setScene(scene);
        stage.show();
    }
```

Рисунок 3.2 – Реалізація з’єднання Java з FXML файлом програми

Для того, щоб відбувся зв'язок з FXML файлом, також необхідно підключити такі бібліотеки:

1) javafx.application.Application;

2) javafx.fxml.FXMLLoader;

3) javafx.scene.Scene;

4) javafx.stage.Stage.

Далі заносимо FXML файл, який являється інтерфейсом програми в окремий пакет проекту (рисунок 3.3) .

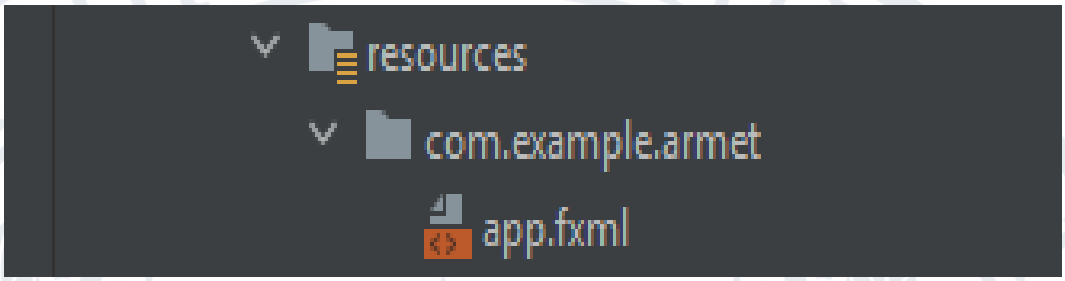


Рисунок 3.3 – Пакет з файлами fxml

3.5 Робота з програмним продуктом

Візуально інтерфейс додатку простий і зрозумілий, не вимагає додаткових інструкцій по експлуатації. Всього в додатку буде два вікна: перше вікно – це головне меню, інтерфейс якого простий і зрозумілий, він має монітор на якому можна слідкувати за роботою циклу, дві стандартних кнопки у верхньому правому кутку вікна, а також 3 кнопки які мають наступні властивості (Рис 3.4):

1. START – кнопка яка розпочинає роботу циклу, після першого увімкнення або ж після увімкнення STOP.
2. SELECT – кнопка котра відкриває додаткове вікно для налаштувань.
3. STOP – кнопка, котра повністю вимикає роботу циклу.

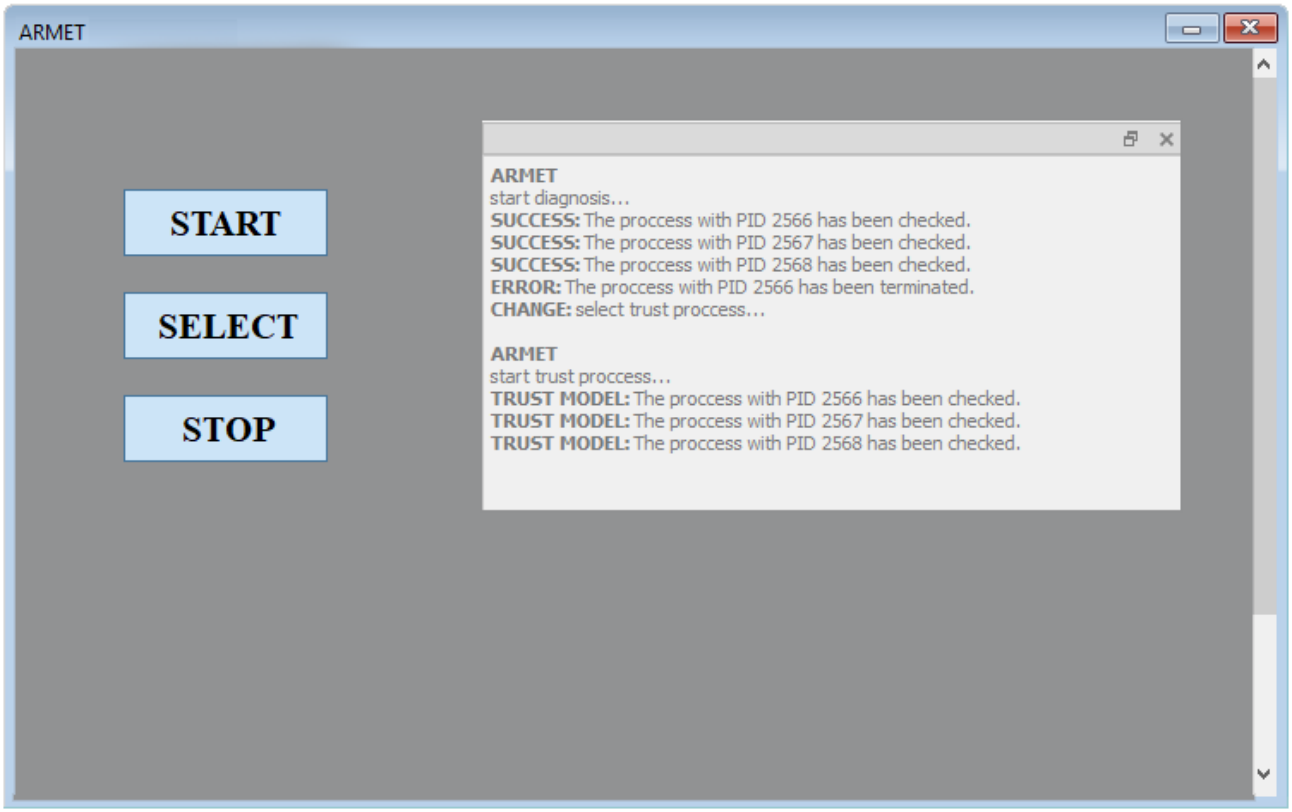


Рисунок 3.4 – Головне вікно додатку

Також, ми маємо друге вікно, яке з'являється після натиснення кнопки SELECT. Це невелике вікно налаштувань, які ми можемо обирати (Рис 3.5):

FULL BACKUP - повернення до попереднього чистого стану

TRUST MODEL – використання моделі довіри

AUTO SAVE – автозбереження коректної роботи циклу, яке відбувається раз на 30 секунд.

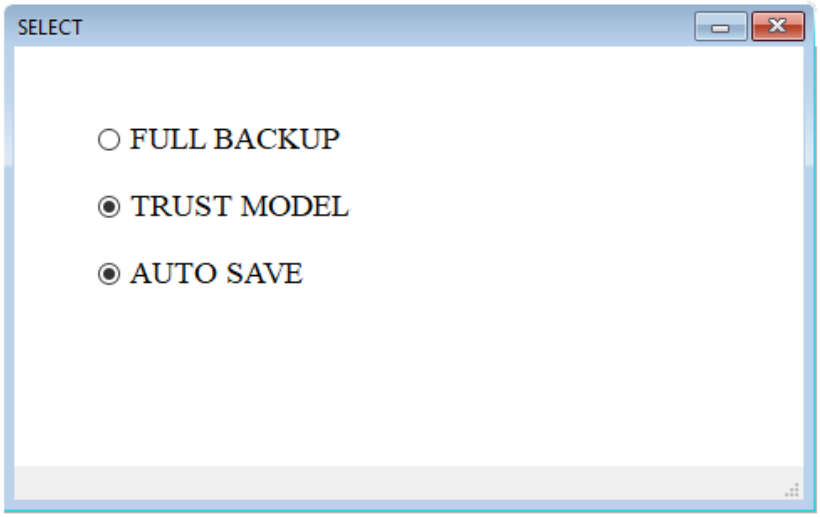


Рисунок 3.5 – Налаштування додатку

3.6 Висновки за розділом 3

В даному розділі було реалізоване створення додатку за допомогою технології JavaFX. Також поступово описано створення програмного забезпечення для використання, а також перевірено графічний інтерфейс для користувача.

ВИСНОВКИ

В даній роботі розглядалась проблематика надійності Інтернету речей з точки зору класичного підходу до оцінки надійності. В результаті якої вдалось знайти ознаки, за якими можна уникнути та захиститись від кібератак та небажаних порушень роботи системи, а також можливість відновлення даних за допомогою технологій захисту інформації, ARMET та моделі довіри, яку вона включає. Розроблено програмне забезпечення на його основі, результати якого базуються на RSM, який дає змогу уникати небажаних дій в системі, а також незалежному PID-контролері, що має додаткові властивості захисту, які були підтвердженні дослідженням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sanja Klisarić, “Hacker attacks as a form of syber terrorism”, Jan 2021, DOI: 10.5937/MegRev2102223K, ISBN: 1820-3159.

2. Salim el Khediri, “Secure Internet of thing System for Smart Houses”, May 2022, DOI:10.1016/j.procs.2022.03.057.

3. Jing Wang, “Business-Oriented IoT-Aware Process Modeling and Execution”, September 2021, in Web Information Systems and Applications (pp.747-755), DOI:10.1007/978-3-030-87571-8_65.

4. Konstantinos M. Giannoutakis, “A Blockchain Solution for Enhancing Cybersecurity Defence of IoT”, December 2020, 2020 IEEE International Conference on Blockchain (Blockchain).

5. Інтернет речі (iot) і їх безпека. [Електронний ресурс]. 2020. URL: http://elites-montage.com.ua/internet-rechi-iot-i-yih-bezpeka.

6. Hakar Mohsin Saber, “IoT: Secured and automated house”, October 2017, 017 International Carnahan Conference on Security Technology (ICCST), DOI:10.1109/CCST.2017.8167862.

7. The Industrial Internet of Things Volume G1: Reference Architecture / [L. Shi-Wan, M. Bradford, D. Jacques та ін.]., 2017. – 58 с. – (IIC:PUB:G1:V1.80:20170131).

8. Aditya Virat, “Analysis and Controlling of Distribution Transformer Parameter using AVR Microcontroller IoT System”, September 2022, Smart Technologies for Power and Green Energy (pp.267-280), DOI:10.1007/978-981-19-2764-5_22.

9. Ali G. Hoseini, J. Tookey, H. Omrani. The essence of smart homes: Application of intelligent technologies towards smarter urban future // BIM, Smart/Intelligent Buildings & the Built Environment – 2016.

10. Rajan Chattamvelli, “A Symmetric Scheme for Securing Data in Cyber-Physical Systems/IoT Sensor-based Systems based on AES”, August 2022, International Journal of Computer Applications 184(24):12-17, DOI:10.5120/ijca2022922278.

11. Kazukuni Kobara, “Cyber Physical Security for Industrial Control Systems and IoT”, April 2016, IEICE Transactions on Information and Systems E99.D(4):787-795, DOI:10.1587/transinf.2015ICI0001.

12. J. Siegel and S. Sarma, "A Cognitive Protection System for the Internet of Things," in IEEE Security & Privacy, vol. 17, no. 3, pp. 40-48, May-June 2019, doi: 10.1109/MSEC.2018.2884860.

13. Cornelia Győrödi, Robert Győrödi, Roxana Sotoc, “A Comparative Study of Relational and Non-Relational Database Models in a Web- Based Application” in International Journal of Advanced Computer Science and Applications, vol. 6, No. 11, pp78-82, 2015, DOI: 10.14569/IJACSA.2015.061111.

14. Muhammad Taimoor Khan, Dimitrios Serpanos, Howard Shrobe, “Sound and Complete Runtime Security Monitor for Application Software” in arXiv, pp 1-3, 17 Jan 2016, doi: 10.48550/ arXiv:1601.04263v1.

15. P. Velmurugan, “An advanced and effective encryption methodology used for modern IoT security”, April 2021, Materials Today: Proceedings, DOI:10.1016/j.matpr.2021.03.424.

16. Ahed J Abugabah, Luke Houghton, Ahmad Ali, “RFID adaption in healthcare organizations: An integrative RFID adaption in healthcare organizations” in Computers, Materials and Continua, pp 1337-1338, January 2021, DOI:10.32604/cmc.2022.019097.

17. Mohd Badrulhisham Ismail, “Control and Monitoring Air Conditioner: Perspective of Internet of Things”, July 2020, Test Engineering and Management 83(march/april):8215-8220.

18. Aaron Kans, “Java Methods” in Programming in Two Semesters, pp.339-355, October 2022, DOI:10.1007/978-3-031-01326-3_15.

19. Aaron Kans, “Java Case Study”, in Programming in Two Semesters, pp. 605-650, October 2022, DOI:10.1007/978-3-031-01326-3_24.

20. Binildas Christudas, “MySQL”, Practical Microservices Architectural Patterns (pp.877-884), June 2019, DOI:10.1007/978-1-4842-4501-9_27.

21. Jesper Wisborg Krogh, “MySQL 8 Query Performance Tuning”, (pp.199-226), March 2020, DOI:10.1007/978-1-4842-5584-1_11.

22. Ying Bai, “SQL Server Database Programming with Java”, (pp.71-96), January 2022, DOI:10.1007/978-3-031-06553-8_3.

23. Sangjin Lee, Jeewon Jang, Doowon Jeong, “The Recovery Method for MySQL InnoDB Using Feature of IBD Structure”, pp. 59-66, February 2017, DOI:10.3745/KTCCS.2017.6.2.59.

24. A. Salisu, Aminu Bugaje, A. Z. Loko, “IOT BASED HOUSEHOLD ELECTRICITY ENERGY MONITORING AND CONTROL” in FUDMA Journal of Sciences, Vol. 4 No. 4, December, 2020, pp 77 – 84, doi: 10.33003/fjs-2020-0404-466.

25. Howard Shrobe, Dimitrios Serpanos, Muhammad Taimoor Khan, “ARMET: Behavior-Based Secure and Resilient Industrial Control Systems” in Proceedings of the IEEE, PP(99):1-15, doi: 10.1109/JPROC.2017.2725642.

26. Scene Builder [Електронний ресурс] Режим доступу: https://code.m-akery.c-h/uk/library/javafx-tutorial/part1/

27. JavaFX Tutorial: FXML and SceneBuilder [Електронний ресурс] Режимдоступу: https://www.vojtechruzicka.com/javafx-fxml-scene-builder/

28. JavaFX: A Rich Internet Application (RIA) Development Platform [Електроний ресурс] Режим доступу: https:// www.opensourceforu.com/ 2016/08/javafx-rich-internet-application-ria-development-platform/

29. What is an API? [Електронний ресурс] Режим доступу: https://aws.amazon.com/what-is/api/?nc1=h_ls

30. CSS [Електроний ресурс] Режим доступу: https://css.in.ua/

31. UI DESIGN [Електронний ресурс] Режим доступу: https://xd.adobe.com/ideas/process/ui-design/

Додаток А

Код програми

```
boolean codeArmet = false;
        boolean fullProgram = false;
        boolean isAtack = false;

        Object armetName = null;
        Object name1 = armetName;
        int estimate = 0, UserStory;
        Pair applicationStory = null;
        Pair userStory1 = applicationStory;{
          if (applicationStory == null) {
            return null;
           } else {

            Object id = null;
            Bug bug = new Bug(id, armetName, estimate, applicationStory);
            bug.name = armetName;
            bug.estiate = estimate;
            bug.userStoryId = applicationStory.id;

            return bug;
```

```
    }
}
if (codeArmet == fullProgram) {
    int i;
    for (i = 0; i < codeIndex; i++) {
        char c = armetBlock(i);
        FileReader fr = fr1;
        fr = fr1;
        try {
            File file = new File("Armet.java");
            fr = new FileReader(file);
            char[] a = new char[lngth];
            fr.read(a);
            for (char h : a) {
                System.out.print(c);
            }
        } catch (IOException e) {
            e.printStackTrace();
        } finally {
            try {
                fr.close();
            } catch (IOException ex) {
                ex.printStackTrace();
```

```
        }
    }
}
FileReader fr = fr1;
try {
    File file = new File("Armet.java");
    fr = new FileReader(file);
    char[] a = new char[lngth];
    fr.read(a);   // reads the content to the array
    for (char c : a)
        System.out.print(c);   // prints the characters one by one
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        fr.close();
    } catch (IOException ex) {
        ex.printStackTrace();
    }
}

void testRegularPositiveCaseWithUserStories; {
    int nextId = 1;
```

```
        Sprint sprint = new Sprint(40, 3);

        assertTrue(sprint.addUserStory(new armetStory(nextId++,
count)));
        assertTrue(sprint.addUserStory(new armetStory(nextId++,
count)));
        assertTrue(sprint.addUserStory(new armetStory(nextId++,
count)));

        Ticket[] tickets = sprint.getArmet();

        assertEquals(3, tickets.length);
        assertEquals(tickets[0].toString());
        assertEquals(tickets[1].toString());
        assertEquals(tickets[2].toString());

        assertEquals(40, sprint.getTotalEstimate());
    }

}
```

```
package com.example.patientcontrol;

import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;
public class DatabaseHandler extends Configs {
    Connection dbConnection;
    public Connection getDbConnection() throws ClassNotFoundException,
SQLException{
        String connectionString = "jdbc:mysql://" + dbHost + ":" + dbPort + "/"
+ dbName;
        Class.forName("com.mysql.jdbc.Driver");
        dbConnection = DriverManager.getConnection(connectionString,
dbUser, dbPass);
        return dbConnection;
    }
}
<?xml version="1.0" encoding="UTF-8"?>
        <?import javafx.scene.control.Label?>
        <?import javafx.scene.control.TextArea?>
        <?import javafx.scene.control.TextField?>
        <?import javafx.scene.layout.AnchorPane?>
        <?import javafx.scene.text.Font?>
<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity"
```

```
            minWidth="-Infinity" prefHeight="981.0" prefWidth="700.0"
style="-fx-background-color: #2e3348;" xmlns="http://javafx.com/javafx/17"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="com.example.demo.ControllerHome">
        <children>
          <AnchorPane layoutY="98.0" prefHeight="757.0" prefWidth="700.0"
style="-fx-background-color: #E0FFFF;">
            <children>
              <Label layoutX="79.0" layoutY="18.0" prefHeight="720.0"
                 prefWidth="545.0" style="-fx-background-color: FFFFFF;" />
              <Label fx:id="exitButton" layoutX="157.0" layoutY="638.0"
                 prefHeight="64.0" prefWidth="448.0" text="Menu"
                 textFill="#727272">
                <font>
                  <Font name="Agency FB" size="28.0" />
                </font>
              </Label>
              <ImageView fx:id="exitImage" fitHeight="51.0" fitWidth="49.0"
                  layoutX="101.0" layoutY="646.0" pickOnBounds="true"
preserveRatio="true">
                <image>
                  <Image url="@../../../../java/assets/exit.png" />
```

```
        </image>
    </ImageView>
    <Label layoutX="182.0" layoutY="36.0" prefHeight="71.0"
    prefWidth="390.0" text="Admin">
    <font>
        <Font size="28.0" />
    </font>
</Label>
<Label layoutX="158.0" layoutY="304.0" prefHeight="64.0"
    prefWidth="96.0" text="Status:">
    <font>
        <Font name="Agency FB" size="28.0" />
    </font>
</Label>
<Label layoutX="158.0" layoutY="582.0" text="START"
    textFill="#727272">
    <font>
        <Font size="28.0" />
    </font>
</Label>
<Label layoutX="182.0" prefHeight="71.0" prefWidth="390.0"
    text="SELECT">
```

```
            <font>
                <Font size="28.0" />
            </font>
        </Label>
        <Label layoutX="156.0" layoutY="390.0" text="STOP">
            <font>
                <Font size="28.0" />
            </font>
        </Label>
        <Label layoutX="156.0" layoutY="447.0" text="PAUSE">
            <font>
                <Font size="28.0" />
            </font>
        </Label>
        <TextField layoutX="257.0" layoutY="313.0" prefHeight="46.0"
                   prefWidth="294.0" style="-fx-background-color: #7FFFD4;"
text="Armet window">
            <font>
                <Font size="22.0" />
            </font>
        </TextField>
        <Label layoutX="156.0" layoutY="510.0">
```

```
            <font>
               <Font size="28.0" />
            </font>
         </Label>
         <TextArea layoutX="269.0" layoutY="391.0" prefHeight="38.0"
               prefWidth="338.0" />
         <TextArea layoutX="269.0" layoutY="448.0" prefHeight="38.0"
               prefWidth="338.0" />
         <TextArea layoutX="381.0" layoutY="511.0" prefHeight="37.0"
               prefWidth="225.0" />
         <ImageView fx:id="exitImage1" fitHeight="51.0" fitWidth="49.0"
                layoutX="107.0" layoutY="573.0" pickOnBounds="true"
preserveRatio="true">
            <image>
               <Image url="@../../../../java/assets/visit.png" />
            </image>
         </ImageView>
         <ImageView fitHeight="276.0" fitWidth="267.0" layoutX="227.0"
                layoutY="60.0" pickOnBounds="true" preserveRatio="true">
            <image>
               <Image url="@../../../../java/assets/nopicture.png" />
            </image>
```

```
        </ImageView>
    </children>
    </AnchorPane>
    <Label layoutX="114.0" layoutY="24.0" prefHeight="59.0"
prefWidth="472.0"
        text="Armet control" textFill="LIGHTCYAN">
    </Label>
        </children>
        </AnchorPane>
    <?xml version="1.0" encoding="UTF-8"?>
        <?import javafx.scene.control.Button?>
        <?import javafx.scene.control.Label?>
        <?import javafx.scene.control.FullBackup?>
        <?import javafx.scene.control.TextField?>
        <?import javafx.scene.layout.AnchorPane?>
        <?import javafx.scene.text.Font?>
    <AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-
Infinity"
            minWidth="-Infinity" prefHeight="630.0" prefWidth="700.0"
style="-fx-background-color: #2e3348;" xmlns="http://javafx.com/javafx/17"
            xmlns:fx="http://javafx.com/fxml/1"
            fx:controller="com.example.demo.ControllerArmet">
    <children>
```

```
<AnchorPane layoutY="98.0" prefHeight="530.0" prefWidth="700.0"
style="-
    fx-background-color: #E0FFFF;">
      <children>
        <Label layoutX="79.0" layoutY="37.0" prefHeight="476.0"
            prefWidth="545.0" style="-fx-background-color: FFFFFF;" />
        <TextField fx:id="loginField" layoutX="156.0" layoutY="288.0"
            prefHeight="47.0" prefWidth="420.0" promptText="Логін"
style="-fx-borderradius: 10;">
          <font>
            <Font name="Agency FB" size="25.0" />
          </font>
        </TextField>
        <Label layoutX="241.0" layoutY="56.0" prefHeight="64.0"
            prefWidth="250.0" text="FULL BACKUP" textFill="#727272">
          <font>
            <Font name="Agency FB" size="44.0" />
          </font>
        </Label>
        <Button fx:id="SignUpButton" layoutX="231.0" layoutY="436.0"
        mnemonicParsing="false" prefHeight="51.0" prefWidth="271.0"
style="-fxbackground-color: #f39c63;" text="Registr" textFill="LIGHTCYAN">
        <font>
          <Font name="Agency FB" size="27.0" />
```

```
            </font>
        </Button>
        <PasswordField fx:id="passwordField" layoutX="156.0" layoutY="358.0"
                       prefHeight="47.0"     prefWidth="420.0"     promptText="FULL
BACKUP">
            <font>
                <Font name="Agency FB" size="25.0" />
            </font>
        </PasswordField>
        <TextField fx:id="TRUST MODEL" layoutX="156.0" layoutY="143.0"
                     prefHeight="47.0"     prefWidth="420.0"     promptText="TRUST
MODEL " style="-fx-border-radius:
      10;">
            <font>
                <Font name="Agency FB" size="25.0" />
            </font>
        </TextField>
        <TextField          fx:id="signUpLastNameField"          layoutX="156.0"
layoutY="218.0"
                     prefHeight="47.0"     prefWidth="420.0"     promptText="AUTO
SAVE" style="-fx-borderradius: 10;">
            <font>
                <Font name="Agency FB" size="25.0" />
            </font>
```

```
            </TextField>
        </children>
        </AnchorPane>
        <Label layoutX="114.0" layoutY="24.0" prefHeight="59.0"
prefWidth="472.0"
            text="ARMET" textFill="LIGHTCYAN">
        <font>
          <Font name="Copperplate Gothic Bold" size="52.0" />
        </font>
        </Label>
            </children>
            </AnchorPane>
```