ЗМІСТ

[ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 3](#_Toc515821636)

[ВСТУП 4](#_Toc515821637)

[РОЗДІЛ 1. АНАЛІЗ КОНЦЕПЦІЇ РОЗУМНОГО ТОРГІВЕЛЬНОГО ЦЕНТРУ 5](#_Toc515821638)

[1.1. Концепція розумного торгівельного центру 5](#_Toc515821639)

[1.2. Складові середовища розумного торгівельного центру 5](#_Toc515821640)

[1.3. Безпека в торгівельному центрі 8](#_Toc515821641)

[1.4. Діагностика безпеки середовища розумного ТЦ 11](#_Toc515821642)

[1.5. Аналіз вимог 13](#_Toc515821643)

[Висновки до розділу 14](#_Toc515821644)

[РОЗДІЛ 2. ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ СИСТЕМИ ДІАГНОСТИКИ БЕЗПЕКИ СЕРЕДОВИЩА РОЗУМНОГО ТОРГІВЕЛЬНОГО ЦЕНТРУ 15](#_Toc515821645)

[2.1. Специфікація вимог до технологій розробки системи діагностики 15](#_Toc515821646)

[2.1.1. Мова програмування Java 15](#_Toc515821647)

[2.1.2. Фреймворк Netflix Hystrix 16](#_Toc515821648)

[2.1.3. Фреймворк Spring Security 16](#_Toc515821649)

[2.2. Вибір інструментальних засобів розробки системи діагностики 17](#_Toc515821650)

[2.2.1. Середовище розробки IntelliJ IDEA 17](#_Toc515821651)

[2.2.2. Інструмент тестування API Postman 17](#_Toc515821652)

[2.3. Специфікація структури системи діагностики безпеки 18](#_Toc515821653)

[Висновки до розділу 19](#_Toc515821654)

[РОЗДІЛ 3. РОЗРОБКА МОДУЛІВ СИСТЕМИ ДІАГНОСТИКИ 20](#_Toc515821655)

[3.1. Функціональна структура модулів системи діагностики 20](#_Toc515821656)

[3.2. Модуль діагностики програмного забезпечення системи безпеки 21](#_Toc515821657)

[3.3. Модуль діагностики апаратного забезпечення системи безпеки 27](#_Toc515821658)

[3.4. Модуль діагностики системи безпеки серверного обладнання 33](#_Toc515821659)

[3.5. Модуль діагностики системи безпеки мережі та вузлів передачі даних 37](#_Toc515821660)

[Висновки до розділу 42](#_Toc515821661)

[ПЕРЕЛІК ПОСИЛАНЬ 43](#_Toc515821662)

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API –

# ВСТУП

Сучасний рівень розвитку програмних та апаратних засобів, що забезпечують інтеграцію пристроїв в єдине кероване середовище, яке дозволяє автоматизувати виконання повсякденних завдань, знизити рівень помилок при їх виконанні та централізувати керування підсистемами такого середовища, забезпечується широким спектром технологічних рішень та робить їх досить доступними. З розвитком технологій інтернету речей, ростом їх популярності серед користувачів та перевіркою часом надійності їх роботи, знаходяться все більше галузей господарства, що переходять на використання таких технологій.

Так, сучасні торгівельні центри почали активно впроваджувати такі рішення для створення повноцінного розумного середовища. Проте, на відміну від середовища розумного дому, торгівельний центр є на порядок складнішою системою, адже поєднує в собі функціонуючі підсистеми, які розглядаються як окремі середовища, глобальні мережі обміну даними всередині середовища, інтеграцію сторонніх сервісів та систем зберігання даних та багато інших компонентів, які ускладнюють процес їх контролю. Таке середовище розумного ТЦ є об’єктом мого дослідження в рамках даної роботи.

Враховуючи, що торгівельний центр є складною архітектурною та інформаційною системою, в його середовище інтегруються аварійні системи та засоби запобігання й реагування на екстрені випадки. До того ж, системи торгівельного центру оперують великими обсягами персональних даних відвідувачів, що безмовно гостро ставить питання безпеки і передбачає використання програмних засобів її забезпечення. Очевидно, що такі системи безпеки повинні постійно мати найвищий рівень працездатності, а отже знаходитись в режимі постійного діагностування. Саме тому, система діагностики безпеки середовища розумного торгівельного центру є вкрай важливою системою для забезпечення працездатності цілого середовища. Розробка такої системи є метою даної дипломної роботи.

# АНАЛІЗ КОНЦЕПЦІЇ РОЗУМНОГО ТОРГІВЕЛЬНОГО ЦЕНТРУ

## Концепція розумного торгівельного центру

Поняття "розумного торгівельного центру" було введено не так давно. Вперше автоматизацію управління та розробку програмного та апаратного забезпечення для створення "розумного" середовища для торгівельного центру було запропоновано в рамках розробки екосистеми розумного міста Smart City. Сучасне середовище розумного торгівельного центру являє собою сукупність складну систему обладнання, програмного та матеріального забезпечення, що складається із ряду менш комплексних систем (об'єктів), з'єднаних між собою масштабованою системою комунікації. Основними функціями, що виконує таке середовище є:

• Централізує управління об’єктами, надаючи один спільний інтерфейс для керування ними та їх підсистемами;

• Дозволяє збирати та аналізувати дані та метрики з об’єктів та їх підсистем в реальному часі за допомогою програмних засобів, мережевої інфраструктури та клієнтських пристроїв;

• Уніфікує формати обміну даними між об’єктами та підсистемами, створюючи тим самим запас для масштабування середовища;

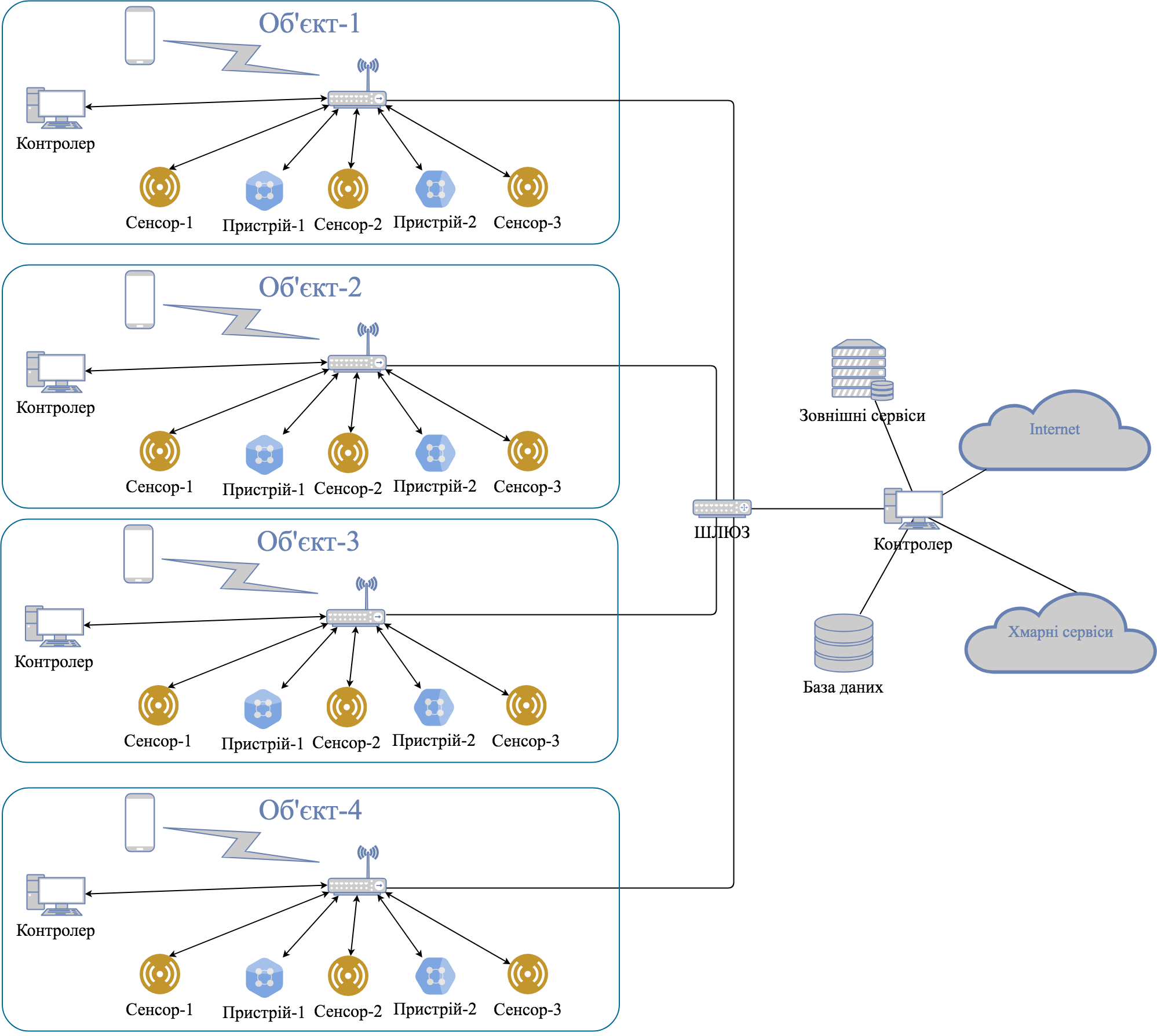
• Впорядковує та фільтрує численні пакети даних та повідомлення, якими підсистеми обмінюються в рамках середовища.

## Складові середовища розумного торгівельного центру

Сучасний торгівельний центр може складатись із кількох кампусів, якими можуть бути окремі будівлі, проте з’єднані в одну інформаційну мережу. Основною структурною одиницею в середовищі розумного ТЦ є об’єкт. Він являє собою магазин або торгову одиницю, в який інтегровано програмні та апаратні засоби для автоматичного функціонування такого об’єкта. Такі засоби під’єднані до центрального контролера об’єкта, який забезпечує обмін даними між пристроями в рамках об’єкта, а також має прямий доступ до:

1. Зовнішніх серверів, що забезпечують роботу та підтримку інформаційних систем окремих об’єктів;
2. Бази даних торгівельного центру;
3. Доступу до мережі інтернет;
4. Інформаційних систем на рівні цілого ТЦ, що забезпечують його функціонування й знаходяться в інших кампусах.

Структурну схему середовища розумного ТЦ наведено на рисунку 1.1.



(рис. 1.1. Структурна схема середовища розумного ТЦ)

Як показано на схемі, об’єкти в середовищі з’єднані з контролером, що відповідає за їх інтеграцію через шлюз, який в свою чергу приймає на вхід пакети даних за протоколом TCP/IP та спрямовує їх до контролера у впорядкованому вигляді. Таке з’єднання забезпечує стійкість системи та дозволяє уникнути втрати даних, адже може будувати черги із запитів, якщо швидкість їх надходження перевищує продуктивність обробки.

В рамках окремого об’єкту функціонують:

* сенсори, призначенням яких є зчитування даних із навколишнього середовища та передача їх на обробку до центрального контролера об’єкту;
* пристрої виконання операцій, команди про які надходять від центрального контролера;
* маршрутизатор, що слугує шлюзом для передачі даних між компонентами всередині об’єкта, а також забезпечує вихід в Інтернет-мережу даного об’єкта для віддаленого керування ним;
* центральний контролер об’єкта, що в режимі реального часу отримує дані із сенсорів, аналізує їх та виконує певні операції засобами передачі команд пристроям об’єкта, а також обробляє запити, що прийшли з-зовні від зареєстрованого користувача на проведення тих чи інших дій з пристроями.

Найбільш популярним видом з’єднання у такого роду об’єктах є бездротове, використовуючи маршрутизатори Wi-Fi, при цьому протокол обміну даними всередині об'єкта є TCP/IP та здійснюється на мережевому рівні передачі даних. Проте, також використовуються й дротові канали передачі даних в рамках об’єкта.

Описуючи повний шлях отримання, обробки та передачі даних в середовищі розумного торгівельного центру, варто виділити такі етапи:

1. Автоматичне або ручне зчитування сенсорами даних про стан навколишнього середовища;
2. Передача даних з сенсорів на обробку до центрального контролера, що здійснюється через маршрутизатор;
3. Обробка даних, отриманих з сенсорів засобами програмного забезпечення, що встановлене на центральному контролері;
4. Передача даних про стан об’єкта на головний контролер торгівельного центру через центральний шлюз, який в свою чергу будує чергу з таких запитів, що надходять з усіх об’єктів середовища, та по черзі направляє їх на обробку до головного контролера;
5. Обробка даних про стани всіх об’єктів середовища, їх аналіз, збереження до бази даних та, при необхідності, комунікація зі сторонніми сервісами для отримання додаткових даних засобами мережі Інтернет.

Охоронні та інші системи в безпеки, що використовуються в торгівельних центрах для контролю входів та виходів, в рамках даної роботи розглядаються як окремі об’єкти, так як мають такий же набір компонентів.

## Безпека в торгівельному центрі

Сучасна система управління торгівельним центром являює собою складне технічне та інфраструктурне рішення, і складається з цілої низки пристроїв, мережевого обладнання та рівнів обробки й передачі даних. Тому, поняття безпеки в ній розглядається як комплекс апаратних та програмних засобів, які підтримують стан захищеності середовища та окремих об'єктів у ньому від внутрішніх та зовнішніх загроз, а також забезпечують стійкість середовища від їх впливу.

Розрізняють такі види безпеки в середовищі розумного торгівельного центру:

* Охоронна безпека;
* Пожежна безпека;
* Інформаційна безпека.

Охоронна безпека ТЦ є комплексом засобів, що застосовується в торгівельному центрі для запобігання випадків крадіжки або терористичних актів та своєчасного реагування, якщо такі випадки мають місце. В даному контексті фізична охорона є хоча і основною, але тільки складовою системи. Ефективна система забезпечення охоронної безпеки поєднує в собі такі елементи:

* фізичну охорону, що виконує завдання відповідно до умов чинного законодавства;
* набір сенсорів та контролерів проникнення, пристроїв сканування та програмного забезпечення для їх функціонування;
* програмне та апаратне забезпечення моніторингу і відеоспостереження.

Очевидно, в рамках автоматизованої системи керування, яким є середовище розумного торгівельного центру, розглядаються два останні елементи, так як вони є технічними засобами та можуть бути інтегрованими до такої системи.

Пожежна безпека є найбільш пріоритетним аспектом безпеки торгівельного центру, так як передбачає постійну ревізію не тільки технічного забезпечення, а й доступності шляхів своєчасної евакуації. В сучасних ТЦ комплекс забезпечення пожежної безпеки розробляється децентралізованим для того, щоб у випадку надзвичайної ситуації та виходу із ладу якоїсь частини обладнання, решта могла забезпечити швидке реагування на пожежу. Не дивлячись на це, на сьогоднішній день розробляються та використовуються автоматизовані системи керування складовими пожежної безпеки ТЦ в цілому. Варто зауважити, що розглядаються системи пожежної безпеки як окремих об’єктів, так і коридорів та службових приміщень торгівельного центру.

До елементів системи пожежної безпеки відносяться:

* сенсори диму, встановлені в об’єктах та інших приміщеннях ТЦ;
* температурні сенсори, що встановлюються в службових приміщеннях, де проводяться роботи або працює обладнання, що призводить до високих температур повітря;
* пристрої водопостачання та гасіння пожежі, що встановлюються в об’єктах та інших приміщеннях згідно з нормами пожежної безпеки;
* пристрої резервної вентиляції для забезпечення протидимового захисту;
* мультимедійні пристрої сповіщення про пожежу (динаміки, мікрофони, світлова та звукова сигналізації);
* програмне та апаратне забезпечення, що відповідає за функціонування блок управління системою пожежної безпеки;

Підсистема оповіщення про пожежу та управління евакуацією в рамках системи пожежної безпеки повинна забезпечувати:

1. подачу звукових і світлових сигналів у всі приміщення;
2. трансляцію мовних повідомлень;
3. трансляцію в окремі приміщення будівлі повідомлень про місце знаходження вогнища пожежі, про шляхи евакуації і дії для забезпечення особистої безпеки;
4. в разі відключення електроенергії - включення евакуаційного освітлення та резервного джерела електроенергії, якщо таке передбачене;
5. зв'язок з усіма кабінетами і приміщеннями, в яких знаходяться особи, які відповідають за безпечну евакуацію.

Інформаційна безпека торгівельного центру є комплексом програмних засобів, стратегій, підходів до управління ризиками та сценаріями реагування на прецеденти, що встановлюється з метою обмеження доступу до конфіденційних даних, якими обмінюються системи та об'єкти в рамках ТЦ для невповноважених осіб.

Система забезпечення інформаційної безпеки середовища розумного ТЦ контролює інформаційну безпеку на таких рівнях:

* захист об’єктів інформаційної системи, до яких належать як локальні, так і глобальні сховища даних, серверне обладнання;
* захист процесів, процедур та програм обробки інформації, до яких належать програмні засоби, що працюють на серверах чи контролерах об’єктів та інше;
* захист каналів зв’язку, включаючи захист інформації в локальній мережі середовища;
* захист інформації в сторонніх сервісах, що використовуються в середовищі для обробки та передачі даних.

Об’єм даних, якими обмінюються системи в середовищі розумного ТЦ, може становити більше 1 Терабайту щодоби, частина з яких – це персональні дані користувачів та клієнтів торгівельного центру. Такі дані мають бути захищені у відповідності до норм, зазначених законодавством. Тому, в рамках середовища розумного ТЦ застосовуються такі засоби захисту інформації:

* системи резервного копіювання та віддаленого зберігання найбільш важливих масивів даних в середовищі на регулярній основі;
* системи резервування всіх ділянок мереж, що відповідають за збереженість даних;
* програмне та апаратне забезпечення для перерозподілення ресурсів мережі у випадках порушення працездатності певних елементів систем або підсистем середовища;
* програмне забезпечення, що забезпечує захист баз даних та іншої інформації від несанкціонованого доступу.

## Діагностика безпеки середовища розумного ТЦ

Під діагностикою безпеки середовища мається на увазі діагностика надійності роботи технічних, апаратних, програмних та функціональних засобів системи безпеки торгівельного центру. Надійність роботи системи безпеки – це властивість технічних систем виконувати задані функції, зберігаючи в часі значення встановлених експлуатаційних показників у заданих межах, що відповідають заданим режимам і умовам використання, технічного обслуговування і зберігання.

Надійність системи безпеки включає в себе такі властивості, як:

* Безвідмовність (властивість системи безупинно зберігати працездатність протягом деякого часу або деякого напрацювання);
* Довговічність (властивість системи зберігати працездатність до настання граничного стану при встановленій системі технічного обслуговування і ремонтів);
* Збережуваність (властивість системи безупинно зберігати справний і працездатний стан протягом і після зберігання; характеризується здатністю об'єкта протистояти негативному впливу умов зберігання на його безвідмовність і довговічність);
* Ремонтопридатність (властивість системи, що полягає в її пристосованості до попередження і виявлення причин виникнення відмов, пошкоджень та усунення їх наслідків шляхом проведення ремонту і технічного обслуговування).

Так як призначенням системи діагностики безпеки є надання своєчасної та актуальної інформації про стан систем безпеки середовища, в даній дипломній роботі розглядаються такі властивості як безвідмовність та збережуваність. Саме ці властивості можуть чітко та в повній мірі відображати рівень працездатності системи безпеки в даний момент часу та є необхідними для прийняття рішень про реагування на зниження цього рівня. Решта властивостей базуються саме на цих двох та можуть розглядатись вже після діагностики.

Для отримання повної інформації про стан працездатності систем безпеки середовища необхідно мати дані про рівень ефективності роботи кожного з елементів систем безпеки, як всередині структурних одиниць, так і обладнання, що забезпечує обмін даними всередині середовища й мережевих каналів передачі даних.

Для цього необхідно проводити динамічне діагностування систем безпеки середовища в цілому та окремих його складових. З огляду на види систем безпеки, що застосовуються в торгівельних центрах, які описані в розділі 1.3, за призначенням варто згрупувати таку діагностику безпеки в 5 класів:

* діагностика безпеки роботи програмного забезпечення окремих об’єктів середовища;
* діагностика апаратного забезпечення, пристроїв, сенсорів та інших елементів, які належать до об’єктів та відповідають за їх автоматизоване функціонування;
* діагностика безпеки серверного обладнання, що забезпечує інтеграцію об’єктів в середовище;
* діагностика мережі передачі даних всередині середовища;
* діагностика роботи сторонніх сервісів, якими користуються в середовищі для отримання додаткових даних.

Серед вище наведених видів діагностики перші два мають проводитись в рамках окремих об’єктів середовища, а решта – в рамках цілого середовища та інформаційних систем, що забезпечують інтеграцію об’єктів та сторонніх сервісів. Кожен з вище названих видів діагностики безпеки потребує спеціальних засобів та окремих модулів системи діагностики. Так, система діагностики середовища розумного торгівельного центру повинна бути розроблена таким чином, щоб ефективно зчитувати, обробляти та аналізувати дані з метрик, з кожної з систем безпеки об’єктів та середовища в цілому.

При діагностуванні систем безпеки, яке є апостеріорним методом аналізу надійності роботи таких систем, головною метою є отримання статистичних оцінок показників надійності технічних, апаратних, програмних та функціональних елементів.

## Аналіз вимог

Отже, для забезпечення постійної перевірки якості роботи системи безпеки в середовищі розумного торгівельного центру, необхідно розробити програмне забезпечення, яке можна було б інтегрувати в роботу середовища для автоматизації моніторингу роботи системи безпеки, діагностування цієї системи та своєчасного реагування у випадках її несправності. В рамках такого програмного забезпечення мають бути реалізовані такі функції:

* стабільна робота з періодичного збору даних з підсистем безпеки;
* уніфікація форматів обміну даними всередині системи діагностики для обробки;
* продуктивна обробка даних про підсистеми безпеки;
* проведення відповідного тестування як програмного, так апаратного й мережевого забезпечення середовища;
* стабільна комунікація з системами середовища, що відповідають за сповіщення про порушення в системах безпеки.

## Висновки до розділу

В даному розділі було наведено концепцію та внутрішню структуру об’єкта дослідження, яким є середовище розумного торгівельного центру, розглянуто складові системи безпеки, що функціонує на кожному з його рівнів, задачі, що виконують такі підсистеми, а також засоби, які для цього застосовуються. Була описана проблема, яку необхідно вирішити в рамках даної дипломної роботи, а також наведені вимоги до програмного забезпечення, що розробляється.

# ТЕХНОЛОГІЇ РЕАЛІЗАЦІЇ СИСТЕМИ ДІАГНОСТИКИ БЕЗПЕКИ СЕРЕДОВИЩА РОЗУМНОГО ТОРГІВЕЛЬНОГО ЦЕНТРУ

## Специфікація вимог до технологій розробки системи діагностики

### Мова програмування Java

Для розробки системи діагностики безпеки середовища розумного торгівельного центру, в якості мови програмування, було обрано Java, що працює на базі віртуальної машини JVM. Такий вибір було зроблено з огляду на ряд вагомих переваг даної мови програмування серед інших для реалізації програмного забезпечення в рамках даної дипломної роботи.

Серед таких переваг, мультиплатформеність – можливість запуску даного ПЗ на будь-якому серверному обладнанні, що має встановлену JVM. Програмний код, написаний на Java компілюється в байт-код, а потім інтерпретується в JVM, на якій він працює в даний момент часу.

Ще однією перевагою є великі можливості до створення багатопотокових програм, які можуть виконувати багато задач одночасно, що дозволяє створювати відлагоджені інтерактивні програми. Це забезпечується широким вибором фреймворків для таких розробок. Один із таких фреймворків Netflix Hystrix використаний в даній програмі та описаний в розділі 2.1.2.

Високий рівень продуктивності роботи програмного коду, розробленого на Java – ще одна перевага даної мови програмування, яка вплинула на вибір для реалізації системи діагностики безпеки. Це досягається тим, що в Java використовується Just-In-Time компілятор.

Не менш вагомою причиною вибору Java є гнучкість даної мови програмування. Це означає, що програмні продукти, розроблені на Java можуть бути легко та швидко адаптованими до змін або доповнення функціоналу. Така властивість досягається широким набором інструментів рефакторингу та доповнення коду.

### Фреймворк Netflix Hystrix

Даний фреймворк був обраний для розробки системи діагностики безпеки середовища розумного торгівельного центру, так як таке середовище є масштабною розподільною інформаційною системою, що складається з багатьох служб та сервісів, які співпрацюють разом.

Такі служби схильні до невдач у виконанні запитів або відкладанні та затримки відповіді. В такому випадку, затримка у відповіді впливає на продуктивність інших сервісів системи та може призвести до недоступності інших частин програми або, у найгіршому випадку, до втрати даних.

Фреймворк Netflix Hystrix був обраний для розробки програмного для запобігання таким випадкам та створення умов еластичості та відмовостійкості сервісів в рамках середовища.

Бібліотека Hystrix допомагає контролювати взаємодію між службами, забезпечуючи «толерантність» до відмов та «толерантність» до затримки. Це покращує загальну стійкість системи шляхом виділення несправних служб і запобігання каскадного ефекту відмов у системі.

### Фреймворк Spring Security

В рамках діагностики безпеки середовища розумного торгівельного центру передбачено проведення тестування програмного забезпечення на різних рівнях. Зокрема, тестування стійкості до загроз обходу аутентифікації клієнтів та сервісів всередині середовища. Для діагностики надійності роботи систем безпеки аутентифікації проводиться тестування з некоректними аутентифікаційними даними, у відповідь на яке система повинна отримати відмову у обробці запиту від сервісів.

Для реалізації такого функціоналу найкраще підходить фреймворк Spring Security, що надає комплексні сервіси безпеки для корпоративних застосунків на базі Java EE для реалізації різних сценаріїв тестування безпеки аутентифікації. Особливий акцент в даному фреймворку робиться на підтримку шаблону Dependency Injection та використанні його реалізації Inversion of Control. Така реалізація робить процес розробки з використанням Spring Security гнучким та ефективним.

До того ж, так як програмний засіб, що розробляється в рамках даної роботи передбачає великі можливості для розширення функціоналу, Spring Security підходить для додатків, які не мають достатньої глибини для типових сценаріїв для корпоративних застосувань. Тому, даний фреймворк забезпечує можливості розробки з використанням десятків корисних функцій безпеки, що можуть бути кастомізованими в залежності від задачі.

## Вибір інструментальних засобів розробки системи діагностики

### Середовище розробки IntelliJ IDEA

Інтегровне середовище розробки IntelliJ IDEA було обрано для процесу розробки програмного забезпечення системи діагностики безпеки середовища розумного торгівельного центру через високий рівень зручності в інтеграції плагінів, фреймворків та сторонніх бібліотек, а також підтримку специфічного синтаксису та компонентів таких бібліотек.

Зокрема, легко інтегруються фреймворки, зазначені в розділі 2.1. Дане середовище зберігає час розробки внаслідок глибокого аналізу контексту та видалення варіантів, що не підходять для використання в конкретному контексті.

### Інструмент тестування API Postman

Postman забезпечує необхідний функціонал для тестування методів API в ході розробки та їх відлогадження. Серед переваг даного інструменту, що застосовувались в процесі розробки системи діагностики безпеки середовища розумного торгівельного центру, можна виділити наявність графічного інтерфейсу для перегляду специфікацію сервісу, що тестується, збереження типових запитів для повторного використання та гнучкість у виборі оточення, для якого проводиться тестування.

## Специфікація структури системи діагностики безпеки

Структура системи діагностики безпеки середовища розумного торгівельного центру являє собою взаємодію між модулями програмного забезпечення, об’єктами середовища й системами його безпеки та класом, що керує запуском в роботу цих модулів. Використовуючи метадані про об’єкти та системи, що підлягають діагностуванню, дані про підключення до них, що визначені в файлах формату XML, клас Runner надсилає запити на виконання сценаріїв діагностики модулям системи.

При отриманні вже проаналізованих даних від модулів, цей клас спрямовує дані до NOTIFICATION SERVICE, що відповідає за відправку інформацію про результати діагностики адміністративному персоналу, що приймає рішення щодо подальшої експлуатації пристроїв чи програмного забезпечення в середовищі в цілому чи в окремих об’єктах.

Структуру системи діагностики наведено на рисунку 2.1.

pictures/STC_Architecture%20(1).pdf

(рис. 2.1. Структура системи діагностики безпеки)

## Висновки до розділу

В даному розділі було проаналізовано та обрано технології, що використовувались для створення ПЗ, а також інструменти, які застосовувалися для розробки та відлагодження цього ПЗ. Було також проаналізовано переваги цих інструментів в контексті розробки системи діагностики безпеки середовища розумного торгівельного центру.

Базуючись на аналізі, проведеному в даному розділі, було визначено такі технології для розробки:

* Мова програмування Java
* Фреймворк Netflix Hystrix
* Фреймворк Spring Security

В якості інструментів обрано:

* Інтегроване середовище розробки IntelliJ IDEA
* Інструмент тестування API Postman

# РОЗРОБКА МОДУЛІВ СИСТЕМИ ДІАГНОСТИКИ

## Функціональна структура модулів системи діагностики

Враховуючи розподіл діагностики безпеки в середовищі розумного торгівельного центру за призначенням, наведений в розділі 1.4, було розроблено такі модулі:

* SOFTWARE\_SAFETY\_DIAGNOSTICS (відповідає за діагностику програмного забезпечення систем безпеки, що використовується об’єктами середовища);
* HARDWARE\_SAFETY\_DIAGNOSTICS (відповідає за діагностику безпеки апаратного забезпечення, сенсорів, контролерів керування та інших пристроїв систем безпеки, що використовується об’єктами середовища);
* SERVERS\_SAFETY\_DIAGNOSTICS (відповідає за діагностування безпеки серверного обладнання середовища розумного торгівельного центру, що забезпечує інтеграцію всіх підсистем та об’єктів середовища);
* NETWORK\_SAFETY\_DIAGNOSTICS (відповідає за діагностику безпеки мережі та каналів передачі даних всередині середовища, а також його комунікацію в мережі Інтернет);

Кожен з вище наведених модулів діагностики використовує додатковий WEB API-модуль DIAGNOSTICS\_API для комунікації із системами безпеки всередині середовища та обмінюється даними, використовуючи REST-архітектуру. Також, розроблено ще один WEB API-модуль NOTOFICATION\_API, призначенням якого є комунікація з сервісами середовища, що відповідають за сповіщення уповноважених осіб про стан системи безпеки торгівельного центру, порушення в його функціонуванні та інших інцидентах.

Структурну схему модулів системи діагностики наведено на рисунку 3.1.

pictures/SAFETY_MODULES_simple.pdf

(рис. 3.1. Структурна схема модулів системи діагностики)

Більш детальну функціональну структурну схему модулів системи діагностики наведено в кресленні \_.

## Модуль діагностики програмного забезпечення системи безпеки

Даний модуль відповідає за апостеріорний аналіз надійності елементів систем безпеки об’єктів торгівельного центру. Для цього проводиться тестування роботи програмного забезпечення на різних наборах даних з певним періодом для виявлення недоліків в його роботі. Головною метою такої діагностики є отримання статистичних оцінок показників надійності елементів систем. Це досягається за рахунок прогнозування часу безвідмовної роботи, виходячи з функції надійності для конкретного елемента.

Вхідною точкою в даному модулі є інтерфейс комунікації з середовищем DiagnosticsHysCmdAPI, що розроблений на основі фреймворку Hystrix, та визначає методи, наведені в таблиці 3.1.

Таблиця 3.1

Структура інтерфейсу DiagnosticsHysCmdAPI

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | Connection | Встановлює з’єднання з системою безпеки об’єкта, ідентифікатор якого отримує як параметр і повертає об’єкт класу Connection |
| executeTest() | ResultTest | Виконує команду запуску запланованих тестів в системі безпеки, використовуючи екземпляр класу Connection та повертає об’єкт ResultTest |

Для реалізації визначених методів в інтерфейсі DiagnosticsHysCmdAPI розроблено клас DiagnosticsHysCmdAPIImpl, що імплементує даний інтерфейс і описує реалізацію відповідних методів, використовуючи вже зазначений фреймворк Hystrix. Структуру даного класу показано в таблиці 3.2.

Таблиця 3.2

Структура класу DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | Connection | Реалізує логіку встановлення з’єднання з системою безпеки об’єкта засобами комунікації з центральним контролером даного об’єкта |
| executeTest() | ResultTest | Запускає в роботу відповідні тести, збирає результати проведення тестів та записує їх до відповідних полів екземпляру класу ResultTest |

До того ж, в даному класі визначено додаткові змінні, що використовуються в роботі даного класу як допоміжні. Особливо вони потрібні для коректного функціонування фреймворку Hystrix. Перелік таких змінних наведено в таблиці 3.3.

Таблиця 3.3

Структура змінних класу DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| CONNECTION\_TIMEOUT | long | Значення часу в мілісекундах, що позначає максимальний термін очікування системи на встановлення з’єднання |
| DEFAULT\_ID | String | Використовується як ідентифікатор системи по замовчуванню |

В методі executeTest() класу DiagnosticsHysCmdAPIImlp реалізується функціонал виконання тестів програмного забезпечення, за допомогою яких збираються необхідні дані про стан системи і на яких базується діагностика. Для цього використовуються нащадки абстрактного класу SS\_Test – SS\_Load\_Test та SS\_Object\_Test. Структуру класу SS\_Test наведено в таблиці 3.4.

Таблиця 3.4

Структура класу SS\_Test

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| test\_id | String | Ідентифікатор тесту, що проводиться |
| test\_Priority | int | Змінна, що відображає рівень пріоритету виконання даного тесту (від 0 до 9, до 0 – найвищий пріоритет); визначається адміністратором, за замовчуванням має середній пріоритет, що рівний 5 |

Даний клас також має один абстрактний метод test(), що повертає сет results типу Set<Double>, який формується на основі результатів тестування та реалізується в його класах нащадках. Перелік методів класу SS\_Load\_Test наведено в таблиці 3.5.

Таблиця 3.5

Структура класу SS\_Load\_Test

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| test() | Set<Double> | Реалізує логіку запуску необхідних методів тестування, наведених нижче та збирає дані про їх результати в сет для подальшої обробки |
| highLoadTest() | Double | Виконує тестування програмного забезпечення з високим навантаженням для отримання даних про надійність роботи системи безпеки в таких умовах |
| lowLoadTest() | Double | Виконує тестування програмного забезпечення з низьким навантаженням для отримання даних про надійність роботи системи безпеки в таких умовах |

Перелік методів класу SS\_Object\_Test наведено в таблиці 3.6.

Таблиця 3.6

Структура класу SS\_ Object\_Test

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| test() | Set<Double> | Реалізує логіку запуску необхідних методів тестування, наведених нижче та збирає дані про їх результати в сет для подальшої обробки |
| testAuth() | Double | Виконує тестування системи авторизації в рамках програмного забезпечення об’єктів середовища для діагностики інформаційної безпеки та збирає дані про результати |
| evaluateCorrectness() | Double | Виконує тестування програмного забезпечення для оцінки коректності обробки даних та їх форматування для діагностики надійності системи безпеки об’єктів |

Важливо зауважити, що для проведення діагностики в рамках методу evaluateCorrectness() необхідно отримати доступ до даних, якими оперує той чи інший об’єкт та заздалегідь отримати очікувані результати виконання від уповноваженої особи.

В результаті виконання вище наведених методів тестування в рамках діагностики безпеки програмного забезпечення в методі executeTest() класу DiagnosticsHysCmdAPIImpl формується екземпляр класу ResultTest, структуру якого наведено в таблиці 3.7.

Таблиця 3.7

Структура класу ResultTest

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| id | String | Ідентифікатор результату діагностики, що проводиться в масштабі об’єкту |
| timestamp | long | Час в мілісекундах проведення даної діагностики |
| results | Map<SS\_Test, Set<Double>> | Мапа результатів тестування в даній діагностиці, де ключ – екземпляр класу SS\_Test або одного з його нащадків, а значення – це сет, наповнений результатами тестування |
| generalResult | double | Загальний результат даної діагностики, що відповідає значенню функції надійності в програмному забезпеченні об’єкта в даний момент часу |

Після отримання результатів проведення діагностики в методі executeTest() класу DiagnosticsHysCmdAPIImpl у вигляді екземпляра класу ResultTest, він передається на обробку до методу evaluateResult() класу ResultChecker, який оцінює результати діагностики, порівнюючи їх очікуваними значеннями.

Після цього, засобами інтерфейсу SS\_HYSTRIX\_NOTOFIER, результати у вигляді інформації про успішність тестування на кожному з елементів ПЗ, відправляються до модулю середовища, що відповідає за сповіщення про результати запланованої діагностики, NOTIFICATION API.

Загальну діаграму послідовності передачі даних в модулі діагностики програмного забезпечення системи діагностики безпеки середовища розумного торгівельного центру наведено в кресленні \_.

## Модуль діагностики апаратного забезпечення системи безпеки

Даний модуль відповідає за тестування надійності роботи апаратних складових охоронної та пожежної систем безпеки в об’єктах. Така діагностика виконується засобами під’єднання до інтерфейсів операційних систем, на яких працюють пристрої та контролери, та отримання актуальної інформації про рівень споживання ресурсів в цих пристроях, завантаженості процесорів, швидкість реагування на запити та всіх інших аспектів роботи апаратного забезпечення, які можна протестувати через центральний контролер об’єкта. Для отримання доступу до метрик апаратного забезпечення, присроїв та сенсорів всередині об’єкта використовується фреймворк Eclipse Kura.

На початку ланцюга збору даних та діагностики стоїть клас Runner, що виконує методи establishConnection() та run(), завданнями яких є встановлення з’єднання з апаратною складовою об’єкта та запуск процедури діагностування.

Початковою точкою отримання даних для обробки в даному модулі є інтерфейс комунікації з середовищем HS\_DiagnosticsHysCmdAPI, що розроблений на основі фреймворку Hystrix, та визначає методи, наведені в таблиці 3.8.

Таблиця 3.8

Структура інтерфейсу HS\_DiagnosticsHysCmdAPI

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | KuraConnection | Встановлює з’єднання з системою безпеки об’єкта, ідентифікатор якого отримує як параметр і повертає об’єкт класу KuraConnection для подальшої комунікації з пристроями об’єкта |
| diagnose() | ResultHardwareDiagnose | Виконує команду запуску запланованих етапів діагностики в системі безпеки, використовуючи екземпляр класу KuraConnection та повертає об’єкт ResultHardwareDiagnose |

Для реалізації визначених методів в інтерфейсі HS\_DiagnosticsHysCmdAPI розроблено клас HS\_DiagnosticsHysCmdAPIImpl, що імплементує даний інтерфейс і описує реалізацію відповідних методів, використовуючи вже зазначений фреймворк Hystrix. Структуру даного класу показано в таблиці 3.9.

Таблиця 3.9

Структура класу HS\_DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | KuraConnection | Реалізує логіку встановлення з’єднання з центральним контролером об’єкта для отримання доступу до апаратного забезпечення об’єкта, що відповідає за функціонування системи безпеки |
| diagnose() | ResultHardwareDiagnose | Запускає в роботу відповідні тести, збирає результати проведення тестів та записує їх до відповідних полів екземпляру класу ResultHardwareDiagnose |

До того ж, в даному класі визначено додаткові змінні, що використовуються в роботі даного класу як допоміжні, аналогічно до подібного класу в модулі діагностики програмного забезпечення. Перелік таких змінних наведено в таблиці 3.10.

Таблиця 3.10

Структура змінних класу HS\_DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| CONNECTION\_TIMEOUT | long | Значення часу в мілісекундах, що позначає максимальний термін очікування системи на встановлення з’єднання |
| DEFAULT\_ID | String | Використовується як ідентифікатор об’єкта по замовчуванню |

В методі diagnose() класу HS\_DiagnosticsHysCmdAPIImlp реалізується функціонал виконання збору необхідної інформації про стан апаратного забезпечення через інтерфейс, наданий центральним контролером об’єкта засобами фреймворку Eclipse Kura, за допомогою яких збираються необхідні дані про стан системи і на яких базується діагностика. Для цього використовуються нащадки абстрактного класу HS\_Checker – HS\_Metrics\_Checker та HS\_Performance\_Checker. Структуру класу HS\_ Checker наведено в таблиці 3.11.

Таблиця 3.11

Структура класу HS\_ Checker

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| test\_id | String | Ідентифікатор діагностики, що проводиться |
| test\_Priority | int | Змінна, що відображає рівень пріоритету виконання даної діагностики (від 0 до 9, до 0 – найвищий пріоритет); визначається адміністратором, за замовчуванням має середній пріоритет, що рівний 5 |
| kuraConnectionProvider | KuraConnection | Об’єкт класу KuraConnection, що забезпечує з’єднання з об’єктами та використовується в діагностуванні |

Даний клас також має один абстрактний метод collectData(), що повертає сет results типу Map<Device, List<Double>>, який формується на основі показників та метрик, отриманих з пристроїв та реалізується в його класах нащадках. Перелік змінних класу HS\_Metrics\_Checker наведено в таблиці 3.12.

Таблиця 3.12

Структура змінних класу HS\_Metrics\_Checker

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| satisfactoryValues | Map<String, Map<Device, List<Double>> | Мапа, ключем якої є ідентифікатор об’єкта, а значенням інша мапа із ключем – екземпляр класу Device (відповідає конкретному пристрою об’єкта), а значенням – список метрик, де перші 2 – референтні значення, останні 2 – актуальні. |
| actualValues | Map<String, Map<Device, List<Double>> | Аналогічна мапа з актуальними значеннями. |

Перелік змінних класу HS\_Performance\_Checker наведено в таблиці 3.13.

Таблиця 3.13

Структура змінних класу HS\_Performance\_Checker

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| satisfactoryPerformance | Map<String, Map<Device, List<Double>> | Мапа, ключем якої є ідентифікатор об’єкта, а значенням інша мапа із ключем – екземпляр класу Device (відповідає конкретному пристрою об’єкта), а значенням – список метрик, де перші 2 – референтні значення, останні 2 – актуальні. |
| actualPerformance | Map<String, Map<Device, List<Double>> | Аналогічна мапа з актуальними значеннями. |

В результаті виконання вище методів збору даних в рамках діагностики безпеки апаратного забезпечення в методі diagnose() класу HS\_DiagnosticsHysCmdAPIImpl формується екземпляр класу ResultHardwareDiagnose, структуру якого наведено в таблиці 3.14.

Таблиця 3.14

Структура класу ResultHardwareDiagnose

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| id | String | Ідентифікатор результату діагностики, що проводиться в масштабі об’єкту |
| timestamp | long | Час в мілісекундах проведення даної діагностики |
| results | Map<HS\_Test, Map<Device, List<Double>> | Мапа результатів тестування в даній діагностиці, де ключ – екземпляр класу SS\_Test або одного з його нащадків, а значення – це сет, наповнений результатами тестування |
| generalResult | double | Загальний результат даної діагностики, що відповідає значенню функції надійності в програмному забезпеченні об’єкта в даний момент часу |
| checkedDevices | Map<Device, Boolean> | Мапа, що зберігає дані про те, чи були перевірені пристрої в ході даної діагностики, де ключем є екземпляр класу Device, а значенням – екземпляр класу-обгортки Boolean. |

В даному класі використовуються об’єкти класу Device, що в свою чергу відображають реальні пристрої об’єктів середовища. Цей клас слугує для інкапсуляції даних про пристрій, належність його до певного об’єкту середовища та може бути розширеним будь-яким необхідним набором мета-даних. Структуру класу Device наведено в таблиці 3.15.

Таблиця 3.15

Структура класу Device

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| id | String | Ідентифікатор пристрою в середовищі |
| name | String | Найменування пристрою, згідно з технічною документацією |
| object\_id | String | Ідентифікатор об’єкту в середовищі торгівельного центру, до якого належить даний пристрій |

Після отримання результатів проведення діагностики в методі diagnose() класу HS\_DiagnosticsHysCmdAPIImpl у вигляді екземпляра класу ResultHaedwareDiagnose, він передається на обробку до методу evaluateResult() класу HS\_ResultChecker, який оцінює результати діагностики, порівнюючи їх очікуваними значеннями.

Після цього, засобами інтерфейсу HS\_HYSTRIX\_NOTOFIER, результати відправляються до модулю середовища, що відповідає за сповіщення про результати запланованої діагностики, NOTIFICATION API.

Загальну діаграму послідовності передачі даних в модулі діагностики апаратного забезпечення системи діагностики безпеки середовища розумного торгівельного центру наведено в кресленні \_.

## Модуль діагностики системи безпеки серверного обладнання

Даний модуль системи діагностики безпеки середовища розумного торгівельного центру відповідає за постійне та своєчасне тестування програмного та апаратного забезпечення серверного обладнання середовища. Варто зауважити, що програмний засіб роботи системи діагностики, що розробляється в рамках даної дипломної роботи, буде працювати на одному з елементів такого серверного обладнання. Саме тому, критично необхідно забезпечити високий рівень надійності роботи системи діагностики серверного обладнання.

Для реалізації відповідного тестування серверного обладнання, що передбачає тестування як програмного, так і апаратного забезпечення такого обладнання, було використано феймворки Eclipse Kura та Spring Security. Таке тестування складається з 2 етапів: збору даних та їх аналіз.

На початку збору даних та діагностики вмикається в роботу клас Runner, що виконує методи establishConnection() та run(), завданнями яких є встановлення з’єднання з апаратною складовою об’єкта та запуск процедури діагностування.

Початковою точкою отримання даних для обробки в даному модулі є інтерфейс комунікації з середовищем SeS\_DiagnosticsHysCmdAPI, що розроблений на основі фреймворку Hystrix, та визначає методи, наведені в таблиці 3.16.

Таблиця 3.16

Структура інтерфейсу HS\_DiagnosticsHysCmdAPI

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getHWConnection(  String id) | KuraConnection | Встановлює з’єднання з системою безпеки об’єкта, ідентифікатор якого отримує як параметр і повертає об’єкт класу KuraConnection для подальшої комунікації з пристроями об’єкта |
| getConnection(String id) | Connection | Встановлює Http-з’єднання з елементами серверного обладнання, приймаючи як параметр ідентифікатор такого елементу, та повертає об’єкт класу Connection |
| diagnoseHardware() | ResultHardwareDiagnose | Виконує команду запуску запланованих етапів діагностики в системі безпеки серверного обладнання, використовуючи екземпляр класу KuraConnection та повертає об’єкт ResultHardwareDiagnose |
| diagnoseSoftware() | ResultTest | Виконує команду запуску діагностики безпеки програмного забезпечення через екземпляр класу Connection, отриманий раніше та повертає об’єкт ResultTest |

Для реалізації визначених методів в інтерфейсі SeS\_DiagnosticsHysCmdAPI розроблено клас SeS\_DiagnosticsHysCmdAPIImpl, що імплементує даний інтерфейс і описує реалізацію відповідних методів, використовуючи вже зазначений фреймворк Hystrix. Структуру даного класу показано в таблиці 3.17.

Таблиця 3.17

Структура класу SeS\_DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | KuraConnection | Реалізує логіку встановлення з’єднання з центральним контролером серверного обладнання для отримання доступу до апаратного забезпечення об’єкта, що відповідає за функціонування системи безпеки |
| getConnection(String id) | Connection | Реалізує логіку встановлення з’єднання з системою безпеки серверного обладнання засобами комунікації з центральним контролером даного об’єкта |
| diagnoseHardware() | ResultHardwareDiagnose | Запускає в роботу відповідні тести, збирає результати проведення тестів та записує їх до відповідних полів екземпляру класу ResultHardwareDiagnose |
| diagnoseSoftware() | ResultTest | Запускає в роботу відповідні тести, збирає результати проведення тестів та записує їх до відповідних полів екземпляру класу ResultTest |

До того ж, в даному класі визначено додаткові змінні, що використовуються в роботі даного класу як допоміжні, аналогічно до подібного класу в модулі діагностики програмного забезпечення. Перелік таких змінних наведено в таблиці 3.18.

Таблиця 3.18

Структура змінних класу SeS\_DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| CONNECTION\_TIMEOUT | long | Значення часу в мілісекундах, що позначає максимальний термін очікування системи на встановлення з’єднання |
| DEFAULT\_ID | String | Використовується як ідентифікатор об’єкта по замовчуванню |

В методі diagnoseHardware() класу SeS\_DiagnosticsHysCmdAPIImlp реалізується функціонал виконання збору необхідної інформації про стан апаратного забезпечення через інтерфейс, наданий центральним контролером об’єкта засобами фреймворку Eclipse Kura, за допомогою яких збираються необхідні дані про стан системи і на яких базується діагностика. Для цього використовуються класи, що належать до модулю діагностики безпеки апаратного забезпечення: нащадки абстрактного класу HS\_Checker – HS\_Metrics\_Checker та HS\_Performance\_Checker.

В подальшому, для виконання необхідного тестування апаратного забезпечення серверного обладнання середовища розумного торгівельного центру, використовуються пакети, класи та інтерфейси з модулю діагностики безпеки апаратного забезпечення, які передають дані до інтерфейсу SeS\_HYSTRIX\_NOTOFIER та засобами методу notify() передаються до NOTIFICATION API.

В методі diagnoseSoftware() класу SeS\_DiagnosticsHysCmdAPIImlp реалізується функціонал виконання тестів програмного забезпечення серверного обладнання середовища розумного торгівельного центру, за допомогою яких збираються необхідні дані про стан системи і на яких базується діагностика. Для цього використовуються класи та інтерфейси модулю діагностики безпеки програмного забезпечення об’єктів середовища: нащадки абстрактного класу SS\_Test – SS\_Load\_Test та SS\_Object\_Test.

В подальшому, для виконання необхідного тестування програмного забезпечення серверного обладнання середовища розумного торгівельного центру, використовуються пакети, класи та інтерфейси з модулю діагностики безпеки програмного забезпечення, які передають дані до інтерфейсу SeS\_HYSTRIX\_NOTOFIER та засобами методу notify() передаються до NOTIFICATION API.

Загальну діаграму класів та послідовності передачі даних в модулі діагностики серверного обладнання середовища розумного торгівельного центру наведено в кресленні \_.

## Модуль діагностики системи безпеки мережі та вузлів передачі даних

Цей модуль відповідає за діагностику надійності роботи мережі та вузлів передачі даних в середовищі розумного торгівельного центру. Така діагностика виконується засобами під’єднання до інтерфейсів операційних систем, що напряму комунікують з вузлами передачі даних, з якими працюють пристрої та контролери. Діагностики системи безпеки мережі та вузлів передачі даних передбачає тестування таких вузлів на відповідність реагування на несанкціоновані запити передачі даних на різних рівнях, проводячи таке тестування різних рівнів на кожному вузлі. Для отримання доступу до даних, необхідних для тестування використовується фреймворк Eclipse Kura.

На початку ланцюга збору даних та діагностики стоїть клас Runner, який має інформацію про з’єднання, необхідні для тестування та виконує методи establishConnection() та run(), завданнями яких є встановлення цих з’єднань запуск процедури діагностування.

Початковою точкою отримання даних для обробки в даному модулі є інтерфейс комунікації з середовищем NS\_DiagnosticsHysCmdAPI, що розроблений на основі фреймворку Hystrix, та визначає методи, наведені в таблиці 3.19.

Таблиця 3.19

Структура інтерфейсу NS\_DiagnosticsHysCmdAPI

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | KuraConnection | Встановлює з’єднання з вузлом передачі даних, ідентифікатор якого отримує як параметр і повертає об’єкт класу KuraConnection для подальшої комунікації |
| diagnose() | NetworkDiagnosticsResult | Виконує команду запуску запланованих етапів діагностики в системі безпеки, використовуючи екземпляр класу KuraConnection та повертає об’єкт NetworkDiagnosticsResult |

Для реалізації визначених методів в інтерфейсі NS\_DiagnosticsHysCmdAPI розроблено клас NS\_DiagnosticsHysCmdAPIImpl, що імплементує даний інтерфейс і описує реалізацію відповідних методів та доповнює додатковими методами, що відповідають рівням тестування, використовуючи вже зазначений фреймворк Hystrix. Структуру даного класу показано в таблиці 3.20.

Таблиця 3.20

Структура класу NS\_DiagnosticsHysCmdAPIImlp

|  |  |  |
| --- | --- | --- |
| **Метод** | **Тип даних, що повертає** | **Опис** |
| getConnection(  String id) | KuraConnection | Реалізує логіку встановлення з’єднання з вузлом передачі даних для отримання доступу до мережевого забезпечення, що відповідає за функціонування системи безпеки |
| diagnose() | NetworkDiagnosticsResult | Запускає в роботу відповідні тести, збирає результати проведення тестів та записує їх до відповідних полів екземпляру класу NetworkDiagnosticsResult |
| highLevelDiagnose() | SimpleNSResult | Запускає в роботу тестування безпечності передачі даних високого рівня запитів у вузлах |
| midLevelDiagnose() | SimpleNSResult | Запускає в роботу тестування безпечності передачі даних середнього рівня запитів у вузлах |
| basicLevelDiagnose() | SimpleNSResult | Запускає в роботу тестування безпечності передачі даних базового (найнижчого) рівня запитів у вузлах |

В методі diagnose() класу NS\_DiagnosticsHysCmdAPIImlp викликаються методи проведення діагностики різних рівнів, що імплементовані в цьому ж класі: highLevelDiagnose(), midLevelDiagnose(), basicLevelDiagnose(). Виконуються вони засобами реалізованих методів у класах HighLevelDiagnostics, MidLevelDiagnostics, BasicLevelDiagnostics відповідно.

Для виконання таких діагностик розроблено клас NonAuthorizedMockClient, що інкапсулює відповідні типи запитів до вузлів мережі передачі даних в середовищі розумного торгівельного центру на регламентованих рівнях. Структуру класу NonAuthorizedMockClient наведено в таблиці 3.21

Таблиця 3.21

Структура класу NonAuthorizedMockClient

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| highLevelRequest | HttpServletRequest | Сформований запит, що надсилається до вузлів мережі передачі даних в середовищі та не використовує ніяких авторизаційних даних |
| midLevelRequest | HttpServletRequest | Сформований запит, що надсилається до вузлів мережі передачі даних в середовищі та використовує CLIENT\_CERT\_AUTH як тип авторизації та визначає параметри авторизації, що не відповідають таким, що вузли даних можуть обробляти |
| basicLevelRequest | HttpServletRequest | Сформований запит, що надсилається до вузлів мережі передачі даних в середовищі та використовує BASIC\_AUTH як тип авторизації з повним набором параметрів авторизації, що не відповідають таким, що вузли даних можуть обробляти |

В результаті роботи відповідних методів класу NS\_DiagnosticsHysCmdAPIImlp формуються екземпляри класу SimpleNSResult, що відображають успішність проведення тестування вузлів мережі, показуючи як вони відпрацювали на несанкціоновані запити. До полів цього класу належать errorCode, в якому зберігається статус відпрацювання запиту та result типу Boolean, що рівний true, якщо результат позитивний (вузол коректно відреагував на несанкціонований запит), та false - у випадку, якщо вузол прийняв в обробку запит, що сигналізує про проблеми у системі аутентифікації в мережі середовища.

Як наслідок, метод diagnose(), що викликає виконання діагностики по рівням, повертає екземпляр класу NetworkDiagnosticsResult, що інкапсулює дані про діагностику одного вузла. Структуру класу NetworkDiagnosticsResult наведено в таблиці 3.22.

Таблиця 3.22

Структура змінних класу NetworkDiagnosticsResult

|  |  |  |
| --- | --- | --- |
| **Змінна** | **Тип даних** | **Опис** |
| id | String | Ідентифікатор тестування вузла |
| timestamp | long | Час в мілісекундах проведення даної діагностики |
| result | Boolean | Змінна, що відображає успішність проведеної діагностики в цілому. Рівна false, якщо діагностований вузол виявився незахищеним хоча б на одному з рівнів |
| allResults | List<SimpleNSResult> | Список результатів діагностики вузла всіх рівнів |

Після отримання результатів проведення діагностики в методі diagnose() класу NS\_DiagnosticsHysCmdAPIImlp у вигляді екземпляра класу NetworkDiagnosticsResult, він передається на обробку до методу evaluateResult() класу NS\_ResultChecker, який оцінює результати діагностики, порівнюючи їх очікуваними значеннями.

Після цього, засобами інтерфейсу NS\_HYSTRIX\_NOTOFIER, результати відправляються до модулю середовища, що відповідає за сповіщення про результати запланованої діагностики, NOTIFICATION API.

Загальну діаграму послідовності передачі даних в модулі діагностики безпеки мережі передачі середовища розумного торгівельного центру наведено в кресленні \_.

## Висновки до розділу

В даному розділі було описано функціональну структуру модулів системи діагностики, компоненти й складові програмного забезпечення, що розробляється в рамках даної роботи.

# ПЕРЕЛІК ПОСИЛАНЬ