Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження ефективності методів і моделей резервного копіювання та аварійного відновлення серверних даних з використанням хмарного середовища
(тема)

Виконала: студентка 2 курсу, групи ІПЗм-18-2
спеціальності 121-Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-наукової програми Інженерія програмного забезпечення
Ніконова А.В.

Керівник проф. Лєсна Н.С.

Допускається до захисту

Зав. кафедри З.В.Дудар

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук

Кафедра програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121–Інженерія програмного забезпечення
(код і повна назва)
Освітньо-наукова програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Ніконовій Анні Вадимівні
(прізвище, ім'я, по батькові)

1.Тема роботи:Дослідження ефективності методів і моделей резервного копіювання та аварійного відновлення серверних даних з використанням хмарного середовища
затверджена наказом по університету від «27 » березня2020р № 473Ст.
2. Термін подання студентом роботи до екзаменаційної комісії «22» 05 2020 р.

3. Вихідні дані до роботи: *методи і моделі резервного копіювання даних, SAPHybrisCommerce 6.7, об'єктно-орієнтована мова програмування Java,AOP, середовище розробки IntelliJIDEA.*
4. Перелік питань, що потрібно опрацювати в роботі: *аналіз предметної галузі і постановка задачі,аналіз програмних засобів резервного копіювання і відновлення даних у Hybris, критерії і методика оцінювання ефективності, опис програмної системи, оцінювання ефективності резервного копіювання та післяаварійного відновлення з використаням обраних методів.*
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій. *Прогнозовані потреби в обсягах сховищ даних, взаємозв'язок між RTO та RPO, комунікація між складовими платформи через RESTfulAPI, багатокластерне середовище в SAPHybris, діаграма розгортання програмної системи Standby, діаграма класів розширення Standbycore, графік залежності RTOвід RPO, графік залежності відсотка завантаженості процесора відRPO, графік залежності кількості звернень до БД від RPO.*

6 Консультанти розділів роботи

Найменування Розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спецчастина	проф. Лєсна Н.С.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів	Примітка
1	Аналіз предметної галузі та постановка задачі	30 березня 2020р.	
2	Розробка методики та моделі	6 квітня 2020р.	
3	Оформлення результатів дослідження	20 квітня 2020р.	
4	Підготовка пояснювальної записки	27 квітня 2020р.	
5	Підготовка презентації та доповіді	2 травня 2020р.	
6	Перевірка роботи на антиплагіат	13 травня 2020р.	
7	Нормоконтроль	15 травня 2020р.	
8	Рецензування	17 травня 2020р.	
9	Занесення атестаційної роботи в електронний архів	18 травня 2020р.	
10	Попередній захист	19 травня 2020р.	
11	Допуск до захисту у зав. кафедри	20 травня 2020р.	

Дата видачі завдання 27 березня 2020 р.

Студент _____
(підпис)

Керівник роботи _____проф. Лєсна Н.С._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до атестаційної роботи: 87 с., 43 рис., 3 табл., 3 додатки, 24 джерела.

ПІСЛЯАВАРІЙНЕ ВІДНОВЛЕННЯ, DRAAS, ЕФЕКТИВНІСТЬ, ЕЛЕКТРОННА КОМЕРЦІЯ, АКСЕЛЕРАТОР, JAVA, HYBRIS, SPRING, AOP

Об’єктом дослідження є процес резервного копіювання та післяаварійного відновлення серверних даних.

Метою роботи є виявлення найбільш ефективних методів і моделей резервного копіювання та післяаварійного відновлення серверних даниху платформі SAPHybrisCommerce.

У результаті роботи розроблена методика та критерії оцінювання ефективності методів післяаварійного відновлення. Розроблена програмна система моделювання та оцінювання ефективності обраних методів, моделей і технологій резервного копіювання.

DISASTER RECOVERY, DRAAS, EFFICIENCY, E-COMMERCE, ACCELERATOR, JAVA, HYBRIS, SPRING, AOP

The object of the research is the process of server data disaster recovery and failover.

The purpose of the work is to identify the most effective methods of backups creation and system failover in the SAP Hybris Commerce platform.

The result of current research is a disaster recovery and failover methods efficiency estimation software methodology and criteria. A software system for efficiency modeling of the chosen alternative methods has been developed.

ЗМІСТ

ВСТУП

Серед IT-спільноти поширена теза: “Адміністратори поділяються на тих, хто робить бекапи, й на тих, хто вже робить”, де ключовим є слово “вже”. На початку XXI століття в цьому вислові йшося переважно про файли на фізичних носіях. Основною цінністю протягом останніх 20 років стала корпоративна інформація, представлена у вигляді записів баз даних - реляційних, об’єктно-орієнтованих, графових тощо. Саме такі дані є рушійною силою у розвитку багатьох технологій, програмних засобів, фреймворків, пов’язаних із їхнім зберіганням, обробкою та резервним копіюванням. Доцільно перефразувати наведене твердження так: “Власники бізнесу поділяються на тих, хто інвестує в бекапи, та тих, хто вже інвестує”.

Облікові дані користувачів, відомості про замовлення та доставку товарів, платіжні транзакції, метадані комбінуються й можуть досягати петабайтів щоденного трафіку на сервері. Вхідні та вихідні потоки даних вимагають високої швидкості передачі, достатнього обсягу пам’яті й потужного API.

Із корпоративної точки зору, втрата або порушення цілісності інформації може коштувати сотні тисяч доларів у грошовому, часовому й навіть репутаційному еквівалентах. IT-галузь потребує гнучких і водночас надійних методів захисту даних від природних і техногенних катастроф, програмних атак, помилок через людський фактор [1].

Післяаварійне відновлення як послуга або Disaster Recovery as a Service (DRaaS) необхідне для створення відмовостійкої IT-системи. Така система забезпечує безперервну роботу критично важливих ресурсів компанії. DRaaS простіше й вигідніше, ніж окреме аварійне відновлення даних [2].

Багато корпорацій-клієнтів віддають перевагу надійному стеку технологій на базі мови JavaEE. Найпопулярнішою e-Commerce платформою, реалізованою на Java, є SAP Hybris Commerce. Hybris - добре спроєктоване рішення, яке надає нові можливості для B2B та B2C маркетингу. Із використанням Hybris розроблені

такі гігантські інтернет-магазини, як Porsche, Volkswagen, Paco Rabbane, Indesit, Deutsche Post, Nikon, Philips, Carlsberg, Henkel та інші [3]. Платформа позиціонується як універсальне комерційне рішення.

Програмне забезпечення SAP Hybris є багатоканальним і може містити від одного до необмеженої кількості сайтів [4]. Це економічно обґрунтовано, бо різні сайти використовують спільну логіку сервісів, фасадів і навіть контролерів. Рекомендованою практикою є використання кластеру із серверів (нод) для оптимізації серверних ресурсів й балансування навантаження.

Попри величезну кількість шаблонів, які реалізовані у Hybris “з коробки” (out of the box, OOTB), платформа надає вкрай обмежені можливості реплікації та післяаварійного відновлення даних. У залежності від потреб бізнесу, щоразу створюються “з нуля” сервіси експорту й імпорту даних. Часто вони є недостатньо перевіреними. Однак на даний момент не існує сервісів чи утиліт для оцінки ефективності даних методів, порівняння, моніторингу їхніх показників. Планування післяаварійного відновлення в Hybris поки що є несистемним явищем.

Актуальність теми полягає у необхідності систематизації процесу створення плану післяаварійного відновлення для Hybris і вибору найбільш оптимального методу резервного копіювання з урахуванням сучасних метрик.

Тематика роботи безпосередньо пов’язана *з* програмами наукових досліджень кафедри ПІ. Використані набуті знання з навчальних дисциплін і матеріали наукових публікацій з enterprice-програмування, основ баз даних, теорії прогнозування, теорії паралельного обчислення, Big Data, теорії прийняття рішень.

Мета роботи: виявлення найбільш ефективних методів і моделей резервного копіювання та післяаварійного відновлення серверних даних.

Задачі дослідження:

– здійснити аналіз методів, моделей і програмних засобів резервного копіювання та післяаварійного відновлення даних, а також розробити методику і визначити критерії ефективності;

– спроектувати сервіси, фасади та аспекти. Створити доменну модель рішення для платформи Hybris;

– розробити core-модуль з основною логікою програмної системи, initialdata-модуль із вхідними даними, occaddon-модуль з реалізацією кросплатформеного RESTful API та storefront-модуль із UI для управління процесами та графічного відображення результатів;

– провести оцінювання ефективності існуючих методів, моделей і технологій за допомогою розробленого програмного продукту. Розробити рекомендації.

Об’єктом дослідження є процес резервного копіювання та післяаварійного відновлення серверних даних.

Предмет дослідження: методи, моделі і технології резервного копіювання та післяаварійного відновлення серверних даних у SAP Hybris Commerce.

У ході даної роботи задля дослідження ефективності використаних методів та технологій застосовувались теоретичні (наукове моделювання) та емпіричні (експеримент, дослідження, вимірювання) наукові методи.

Наукова новизна. Вперше запропонована методика, обрані критерії для оцінювання ефективності методів резервного копіювання і післяаварійного відновлення даних у Hybris, виявлення оптимальної точки балансу між RPO та RTO.

Практична значимість. Результати оцінювання ефективності та розроблені рекомендації щодо вибору методів резервного копіювання даних дозволять полегшити й систематизувати планування післяаварійного відновлення серверних даних та мінімізувати збитки внаслідок позаштатних ситуацій.

Публікації. За результатами дослідження, проведеного під час виконання атестаційної роботи, опублікована стаття “Дослідження ефективності методів і моделей резервного копіювання та післяаварійного відновлення даних у SAP Hybris Commerce” у міжнародному науковому електронному журналі “ScienceOnline”.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ. ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Сучасні сховища корпоративних даних

Технологічні рішення у сучасних сховищах корпоративної інформації побудовані навколо таких основних викликів:

– зберігання великих обсягів даних;

– їхній захист від аварій та несанціонованого доступу;

– гнучке управління ресурсами.

Перша пробема вирішується шляхом переходу із жорстких дисків (HDD)на твердотільні носії (SSD). Хоча HDD залишаються основним типом носіїв у дата-ценрах завдяки своїй низькій вартості, SSD забезпечують значно швидшу інтеграцію та початкову обробку.

Розв’язання другої проблеми полягає в міграції даних у хмарне сховище. Наявність кількох копій бекапів та архівів даних у хмарах з різними провайдерами та регіонами фізичної геолокації серверів підвищує надійсність та відмовостійкість системи.

Третє питання частково задовольняється за допомогою автоматичних утиліт та аналізаторів даних у Big Data, засобами Machine Learning, здатних класифікувати та кластеризувати дані так, що бізнес-користувачі отримують доступ до найбільш релевантної для них інформації [5].

Згідно із дослідженням Harvard Business Analytics, 2020 рік очікується як період винятково швидкого зростання обсягів даних [1]. Це стосується як активних сховищ - “supply”, тобто тих, які безпосереедньо використовується застосунками, там і резервних - “demand", призначених для аварійного перемикання тощо.

Прогнозовані потреби в сумарній ємності інформаційних накопичувачів у світі [1] наведені на рисунку 1.1.

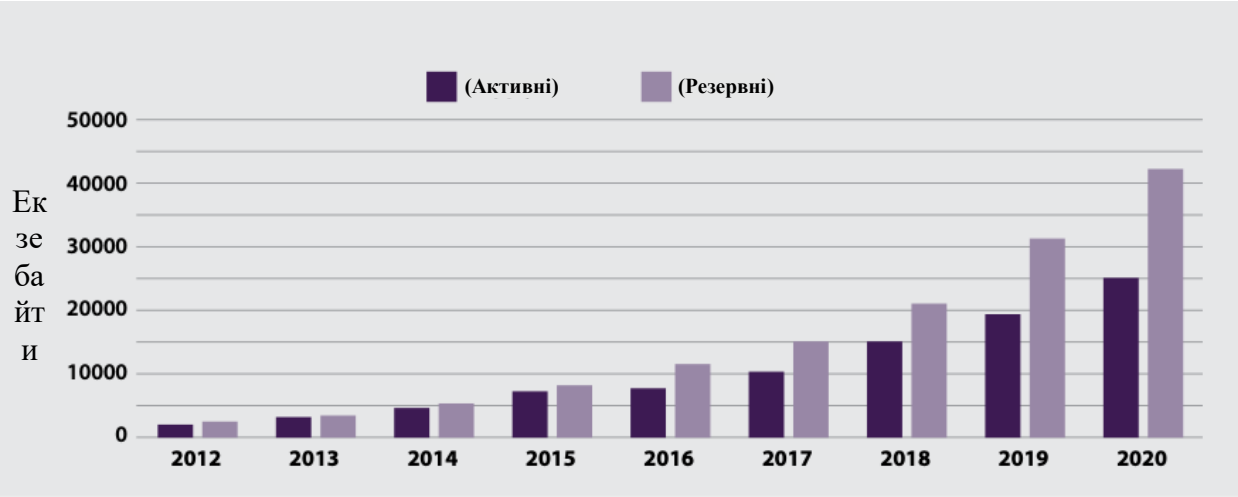

Рисунок 1.1 – Прогнозовані потреби в обсягах сховищ даних

Дослідження Forrsights Hardware Survey, яке проводилося на основі 754 північноамериканських IT-компаній, показало, що стрімкий зріст обсягів сховищ даних пов’язаний переважно із потребами бізнесу. Так, 34% опитуваних зазначили, що потребують більше даних для діяльності, пов’язаною із комерцією. 31% респондентів підвищили обсяги пам’яті, доступної для реплікації та післяаварійного відновлення.

Згідно з іншим опитуванням Forrsights Hardware Survey на базі 100 компаній, найпріорітетнішими вимогами до сховищ даних були визнані ефективність, здатність до післяаварійного відновлення та масштабованість. При цьому 71% опитаних зізналися, що аспектом їхньої поточної інфраструктури зберігання, який потребує вдосконалення чи розширення, є захист даних [6].

1.2 Післяаварійне відновлення даних

Післяаварійне відновлення (Disaster recovery, або DR) - це завдання, згідно з яким компанія інвестує кошти в апаратне та програмне забезпечення, яке буде використане у випадку, якщо внаслідок катастрофи основний сайт чи джерело даних виявиться недоступним.

Сучасні люди звикли до постійно пов’язаного світу, в якому інформація доступна 24/7, і не пробачають незапланованого простою потрібних їм сайтів, незалежно від причини. Не так давно користувачі були більш лояльними до недоступності веб-ресурсу протягом годин або навіть днів, але сьогодні навіть частка години вважається невиправданою: інформація потрібна “тут і зараз”.

До основних загроз для інформаційних ресурсів відносять збої в апаратному забезпеченні, помилки програмного забезпечення, пошкодження даних та катастрофи.

Більшість бізнес-користувачів помилково вважає, що IT-компаніям легко створити стійке до відмов і захищене від стихійних ситуацій середовище. Але це дуже нетривіальна задача.

Традиційні підходи до післяаварійного відновлення включають “гарячий сайт” (hot site),“холодний сайт” (cold site) та“теплий сайт” (warm site) [5].

1.2.1 Підхід “гарячого сайту”

У підході “гарячого сайту” організація повністю дублює своє середовище з основного на альтернативний сервер, який стає базисом для подальшого післяаварійного відновлення. З точки зору інвестицій та обслуговування цей спосіб коштує дуже дорого, а синхронізація серверів та інших компонентів займає багато часу.

Типовий hot site складається із серверів, систем зберігання даних та мережевої інфраструктури, які разом забезпечують логічне дублювання основного сайту. Релізи та оновлення (патчі) на основному та альтернативному серверах є ідентичними. Резервне відновлення може проходити в автоматичному або ручному режимі, залежно від бізнес-потреб та наявних ресурсів.

Організації можуть запускати свої сайти в режимі “активний-активний” або “активний-пасивний". В “активно-активному” режимі програми на первинних

сайтах і на сайтах відновлення постійно працюють, а дані реплікуються двонаправлено. Цей підхід є найбільш високовартісним. У “активно-пасивному” режимі один сайт діє як основний, а дані реплікуються на пасивні сайти.

1.2.2 Підхід “холодного сайту”

Підхід “холодного сайту” передбачає, що вже після аварії організація надсилає носії з бекапами на пусті пристрої, але без гарантії, що нові придбані компютери вчасно прибудуть і коректно підтримуватимуть відновлені програми й дані. Такий процес може займати дні або навіть тижні.

Хоча більшість експертів навіть не розглядаютьcoldsite у контексті DR-планування (через низьку ефективність і високий рівень ризиків), деякі застосунки можуть бути наленим чином відновлені за допомогою цього підходу, якщо це відповідає бізнес-потребам організації.

Ще одна причина, по якій організації вибирають такий ризикований підхід - вони роблять ставку на те, що катастрофа не відбудеться, і таким чином інвестиції не потрібні.

1.2.3 Підхід “теплого сайту”

У підході “теплого сайту” організація використовує компромісне рішення між дорогим “гарячим сайтом” та пустим “холодним сайтом”. Альтернативний сайт може використовувати як власні, так і спільні з основним сайтом сервери. Дані на них можуть бути частково синхронізовані.

Таким чином, післяаварійне відновлення за допомогою warm site відбувається протягом кількох днів і є швидшим, ніж при cold site, однак довшим і дешевшим, ніж із hot site.

Недолік усіх трьох наведених підходів полягає у тому, що вони не відповідають сучасним вимогам щодо економічно вигідного та agile-орієнтованого відновлення. Зазвичай користувачі очікують, що програми будуть запущені протягом частки години. Розроблений правильно, “гарячий сайт” може задовольнити цей попит, але його вартість буде значно вищою за інші аналоги.

У звязку з цим актуальності набуло пісялаварійне відновлення як сервіс (DRaaS), спрямоване на досягнення максимальної відмовостійкості при найменших витратах [5].

1.3 Післяаварійне відновлення як послуга DRaaS

Починаючи з ранніх 2000-х років, багато різновидів сервіс-провайдерів розробили й запровадили цілі індустрії, спрямовані на зниження вартості й складності основних процесів і технологій. Наприклад,програмне забезпечення як послуга (Software as a Service, SaaS), інфаструктура як послуга (Infrastructure as a Service, IaaS), платформа як послуга (Platform as a Service, PaaS), післяаварійне відновлення як послуга (Disaster recovery as a Service, DRaaS).

PaaS базується на апаратному або програмному забезпеченні, що надається іншим публічним постачальником. Це означає, що користувач розміщує свій застосунок, додаткове проміжне програмне забезпечення на ресурсах провайдера. Таким чином, вартість оренди якісної платформи з повною підтримкою і гарантією стабільності й масштабованості значно менша, ніж вартість побудови аналогічної інфраструктури “з нуля”. Це забезпечує швидший вихід на ринок [7]. Водночас слід пам’ятати, що у такому випадку застосунок розгортається за правилами власника платформи.

Згадані “as a service”-рішення створили цілком нові парадигми для використання технології бізнесом [5]. Однією з таких парадигм у галузі PaaS стала платформа SAP Hybris Commerce, у рамках якої проводиться дане магістерське дослідження.

Післяаварійне відновлення даних як послугу DRaaS слід розуміти як сервіс для відновлення даних і забезпечення відмовостійкості системи. У його контексті можуть використовуватися хмарні й локальні ресурси, апаратні та програмні засоби для резервного копіювання важливої інформацій і додатків. Завдяки цьому навіть унаслідок технічного збою бізнес-компанії продовжують працювати у звичайному режимі .

DRaaS гарантує безперервність функціонування критично важливих ресурсів клієнта. До таких ресурсів належать записи баз даних, медіа, сервіси, утиліти тощо. Основна мета DRaaS – можливість майже миттєвого (в ідеалі - миттєвого) перемикання з основного майданчика, який прийнято називати “продуктивним”, на резервний. Це стає можливим завдяки реплікації віртуальної машини [8], або використанню інших платформенно-специфічних методів.

1.3.1 Хмарні ресурси у DRaaS

Послуга DRaaS часто базується на використанні хмарних сховищ, які неперервно удосконалюються. Завдяки ній організаціям легко налаштувати альтернативні сайти для подальшого післяаварійного відновлення. Як і решті “as a service”-пропозицій, професійне програмне забезпечення допомагає післяаварійному відновленню як сервісу спростити весь процес як для компаній будь-яких масштабів, так і для сервіс-провайдерів, які пропонують цю послугу.

Важливість післяаварійного відновлення як сервісу полягає у тому, що воно відображає інноваційні й найвигідніші способи створення резервних копій (бекапів) критичних даних та швидкого відновлення системи після катастрофи.

DRaaS здійснює це за допомогою інфраструктури хмарних ресурсів, використання якої є в рази дешевшим за відповідні за ємністю локальні системи.

Основними перевагами використання хмарних сховищ у післяаварійному відновленні як послузі є можливість масштабування та спільного доступу до ресурсів.

Однією із рекомендацій щодо створення резервних копій є правило “3-2-1”, яке формулюється наступним чином:

– зберігання 3 копій даних: 1 основної, 2 бекапів;

– використання 2 різних типів носіїв;

– розміщення 1 набору даних в хмарі за допомогою Disaster Recovery as a Service [5].

Підсумовуючи, можна зробити висновок, що DRaaS є простою й високоефективною реалізацією післяаварійного відновлення (DR), часто з використанням хмарного середовища.

Для оцінки ефективності DRaaS використовують 2 основні метрики: об’єктивна точка відновлення та об’єктивний час відновлення. Вони є основними концептами, які забезпечують неперервність бізнесу.

1.3.2 Об’єктивна точка відновлення RPO

Об’єктивна точка відновлення (Recovery Point Objective, RPO) відображає толерантність компанії до втрат, тобто обсяг даних, які можуть бути втрачені без отримання значної шкоди для бізнесу. Виражається як час від здійснення останнього бекапу до аварії, що спричинила втрату даних [8].

Якщо організація здійснює частковий або повний бекап даних періодично, наприклад, щодоби, тоді у найнесприятливішому випадку під час аварії вона втратить кількість даних, еквівалентну 24 годинам. Для деяких застосунків це нормально, для інших - неприпустимо.

При 4-годинному RPO максимальний розрив між бекапом і втратою даних становить 4 години. При цьому припускається, що розподіл даних протягом доби є рівномірним. Тобто якщо сайт, який оброблює дані, був востаннє забекаплений опівночі, а “впав” о четвертій ранку, насправді могло бути втрачено мало даних (або й ніяких) через неактивність користувачів.

Якщо сильно завантажений сайт востаннє резервно копіювався о 10 ранку, а аварія сталася через годину,при такому самому RPO = 4 години потенційно могла бути втрачена велика кількість цінної й навіть незамінної інформації. У такому разі слід організувати частіші бекапи так, щоб успіно підібрати RPO, оптимальне для конкретного бізнесу.

Залежно від пріорітету застосунку, індивідуальні плани RPO зазвичай варіюються від 24 до 12, 8, 4 годин, аж до майже-нульових (таких, що вимірюються в секундах). 8-годинні й більші RPO можуть мати перевагу над частішими бекапами завдяки меншому впливу на потужність та завантаженість системи. 4-годинне RPO передбачає копіювання даних за розкладом, майже-нульові обєктивні точки відновлення - неперервне копіювання.

1.3.3 Об’єктивний час відновлення RTO

Об’єктивний час відновлення (RTO, Recovery Time Objective) – це час, протягом якого застосунок може бути недоступним для користувачів без значної шкоди для бізнесу.

Деякі сайти й програми можуть залишатися поза обслуговуванням протягом кількох днів без суттєвих наслідків. Інші ж високопріоритетні застосунки можуть бути недоступними лише кілька секунд, негативний сценарій створює незручності для співробітників компанії, клієнтів та спонсорів, таким чином знижуючи показники бізнесу [5].

RTO – це не просто відрізок часу між втратою та відновленням системи. Об’єктивний час відновлення називається так, бо також враховує кроки, які IT-компанія повинна вжити для відновлення програми та її даних.

Якщо власники IT інвестували кошти у системи відмови (failover), які дозволяють перемикатися на обладнання в режимі очікування за умови несправності основного, тоді RTO може виражатися навіть у кількох секундах. Локальне обладання таким чином може бути відновлене згодом, у той час як відбулося миттєве перемикання і застосунок розгортається й обробляється у хмарі. Такий варіант є досить дорогим у реалізації, бо є втіленням підходу “гарячого сайту”.

1.3.4 Планування RPO та RTO

Обидві метрики RTO та RPO мають такі спільні характеристики:

– є основними критеріями ефективності після аварійного відновлення як послуги;

– потребують ресурсів сервера задля зменшення своїх показників;

– характеризують обсяг даних і вартість втрат, які занає бізнес у випадку порушення цілісності даних;

– вимірюються в одиницях часу;

– встановлюються відповідно до пріорітету застосунку та даних;

– прямують до значення, близького до 0;

– неможливо зебезпечити нульове RTO й RPO.

Взаємозв’язок між об’єктивною точкою відновлення RPO та об’єктивним часом відновлення RTO показаний на рисунку 1.2.

Рисунок 1.2 – Взаємозв’язок між RTO та RPO

Відмінною особливістю між RTO та RPOє їхня мета. Час відновлення RTO пов’язаний із програмами та системами. Показник включає відновлення даних, але переважно описує обмеження в часі на простій програми. Точка відновлення RPO пов’язана із даними, втраченими внаслідок відмови системи.

Задача планування показників RTO і RPO - категоризувати наявні застосунки за пріорітетом і потенційними втратами, оцінити наявні фінансові ресурси для аварійного відновлення.

Можна виділити такі загальні принципи:

– майже-нульові RTO вимагають наявність failover-системи;

– 4-годинні RTO дозволяють досить відновити локальне обладнання, включаючи апаратне та програмне забезпечення, з гарантією повної доступності даних;

– для 8-годинних і більших RTO компаніям слід укладати угоди на технічне обслуговування із спеціалізованими системними інтеграторами.

– для забезпеченя мінімального RTOі відновідно 100% тривалості роботи (uptime) слід інвестувати в failover-системи, які гарантують неперервну реплікацію даних [5].

1.4 Акселератори для електронної комерції

У різних проектах бекенд-розробникам неперервно доводиться розв’язувати однотипні задачі:

– авторизація та автентифікація користувачів;

– численні операції CRUD (створення, зчитування, оновлення та видалення) для сутностей предметної галузі;

– передача об’єктів шару трансферу даних (Data Transfer Object) міжрізними модулями застосунку;

– реалізація транзакцій, додавання товару до кошика, оформлення замовлення (order consignment);

– створення Unit- та інтеграційних тестів і т.д.

Щоб не повторюватися й не перевикористовувати зроблені раніше рішення (принцип “Don’t repeat yourself”, DRY), в кінці минулого століття стали набирати популярність CMS-платформи та акселератори [9]. Акселератори – “прискорювачі”, що містять набір готових програмних реалізацій для основних бізнес-процесів. Вони дозволяють почати розробку не “з нуля”, а зі створеним заздалегідь каркасом, який може містити кілька модулів.

Перевагою акселераторів є зниження вартості й часу розробки, масштабованість програмного забезпечення, уніфікація складних процесів, надійність.

Попри те, що акселераторам певною мірою вдається вирішити проблему повторного використання коду, до їхнього недоліку можна віднести підвищений обсяг памяті, потрібний для розгортання та підтримки застосунку на сервері, порівняно зі стандартним (“неприскореним”) кодом. У подальшому це може бути вирішене за допомогою міграції в хмару і повного використання сервісів хмарних провайдерів.

1.5 Акселератор SAP Hybris Commerce

SAP Hybris – це сімейство продуктів німецької компанії Hybris, яка займається продажем програмного забезпечення для електронної комерції, маркетингу, управління товарами і цінами, продажу й обслуговування клієнтів. Hybris надає рішення, які дозволяють будь-якій організації скоротити витрати, зекономити час, знизити складність і витрачати менше зусиль для досягнення відмінної якості обслуговування клієнтів.

SAP Hybris Commerce Accelerator – програмне рішення для електронної мультиканальної комерції з ефективними шаблонами й інструментами [4]. Базується на такому технологічному стеку: Java 8, Spring, Hybernate, підтримує легке перемикання між базами даних, у тому числі різних провайдерів:

– MySQL;

– Oracle SQL;

– Hyper SQL (HSQL), за замовчуванням;

– PostgreSQL;

– SAP HANA.

Акселератор забезпечує найкращі практики E-commerce і містить значний досвід роботи з клієнтами. Hybris поєднує у собі дві платформи в одному місці:

– бізнес-для-Користувача (Business-to-customer, B2C) - підхід, при якому товари або сервіси продаються безпосередньо кінцевим користувачам для їхнього подальшого використання;

– бізнес-для-Бізнесу (B2B) - спосіб комерційної активності, при якому передбачається посередництво між різними представниками бізнесу.

Найактивніше платформа Hybris використовується у таких галузях, як комерція, бізнес, автомобільна індустрія, освіта, релігія і реклама [10]. Діаграма розподілу кількості Hybris-сайтів за галузями показана на рисунку 1.3.

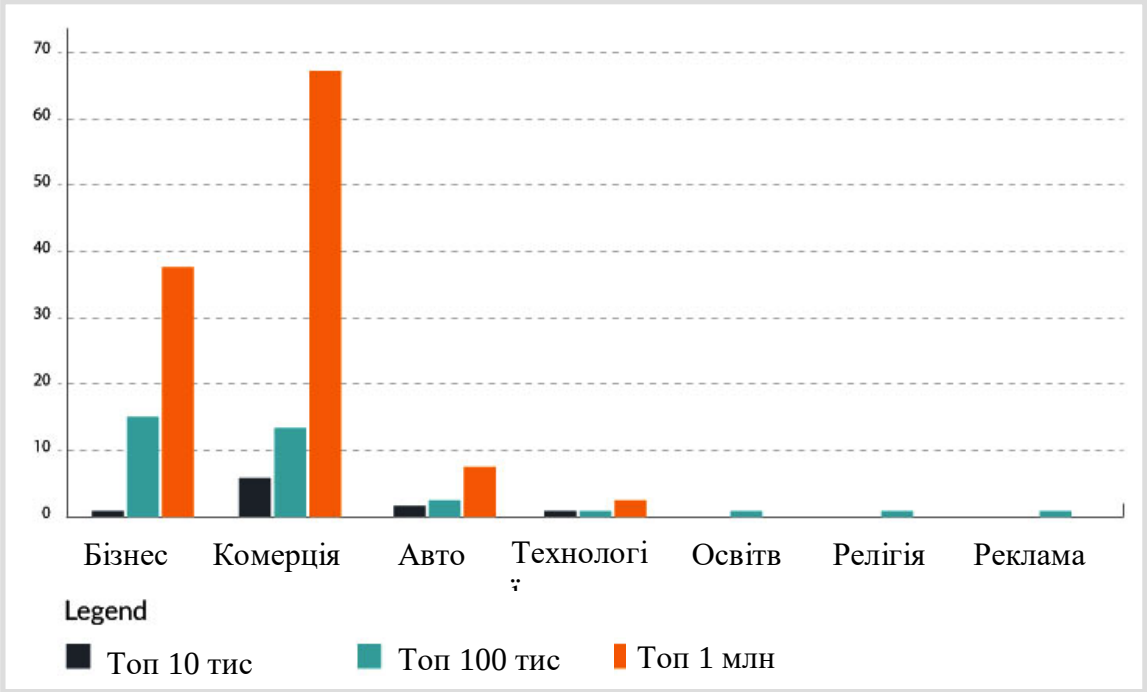

Рисунок 1.3 – Діаграма розподілу кількості Hybris-сайтів за галузями

Hybris є багатоканальною системою, основні елементи якої обмінюються повідомленнями за допомогою RESTfulAPI, як це показано на рисунку 1.4.

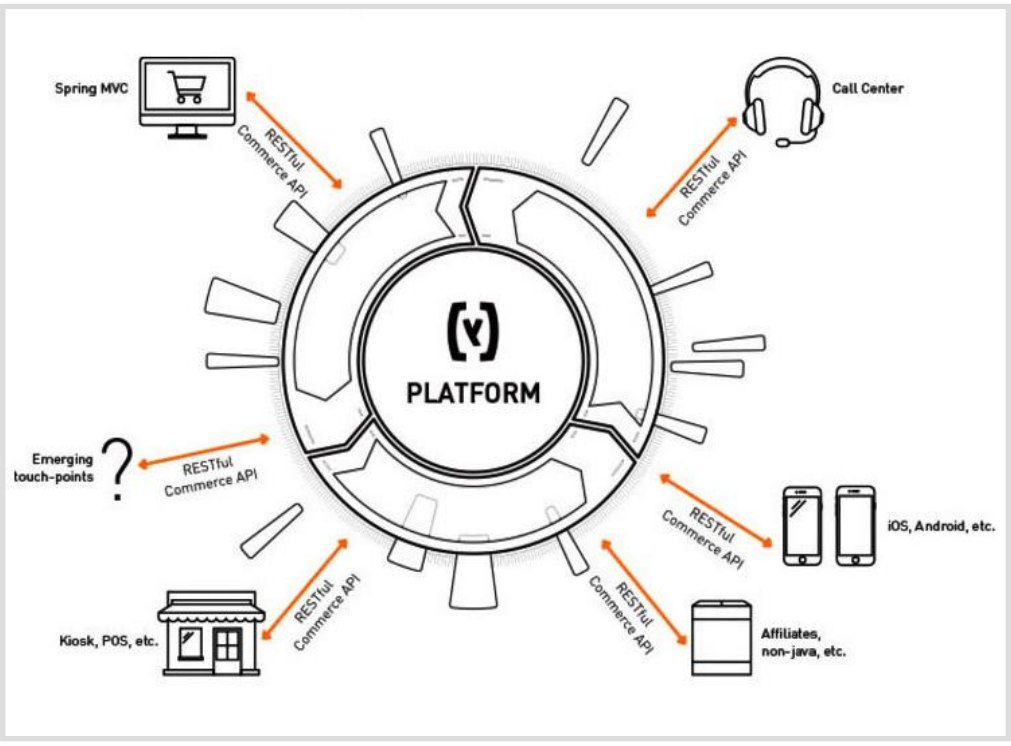

Рисунок 1.4 – Комунікація між складовими платформи через RESTfulAPI

Платформа розроблена для того, щоб упоратися із надвисоким потоком користувачів, а також значними обсягами замовлень, надаючи широкі модливості масштабування коду і швидкого підключення нових компонентів для потреб маркетингу. Таким чином, клієнтам не потрібно витрачати зайві кошти для оновлень і управління новим функціоналом.

Гібридна B2B2C e-Commerce платформа Hybris базується на контейнері Java EE Servlet, використовує середовище Spring – найкращий фреймворк на основі мови Java для корпоративних платформ. Сервер застосунків Apache Tomcat обробляє бізнес-інформацію, необхідну для роботи ядра платформи Hybris [3].

Платформа багата на різноманітні розширення. Модулі, доступні “з коробки” (out of the box, OOTB), можуть бути підключені відповідно до потреб бізнесу. Крім стандартних компонетів (основних програмних розширень - “extensions" та невеликих функціональних додатків - “addons”), користувач має доступ до ресурсу SAP Hybris Market з платними та безкоштовними модулями.

Приклади інтерфейсів сайту, реалізованого на SAP Hybris Commerce, показані на рисунках 1.4, 1.5.

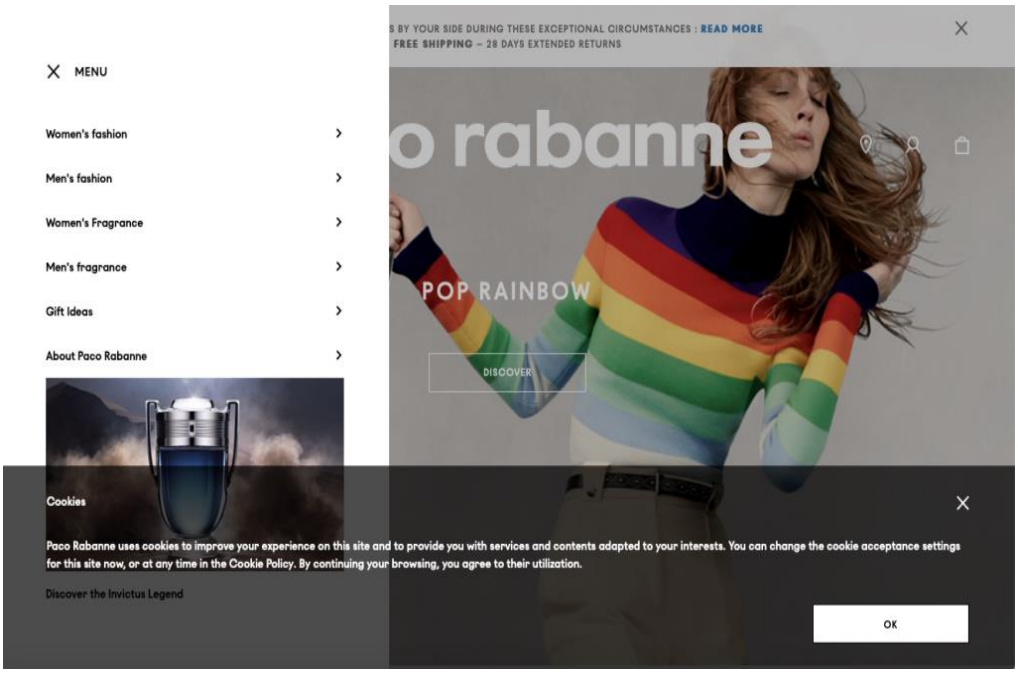

Рисунок 1.4 – Приклад інтерфейсу перегляду категорій у Hybris

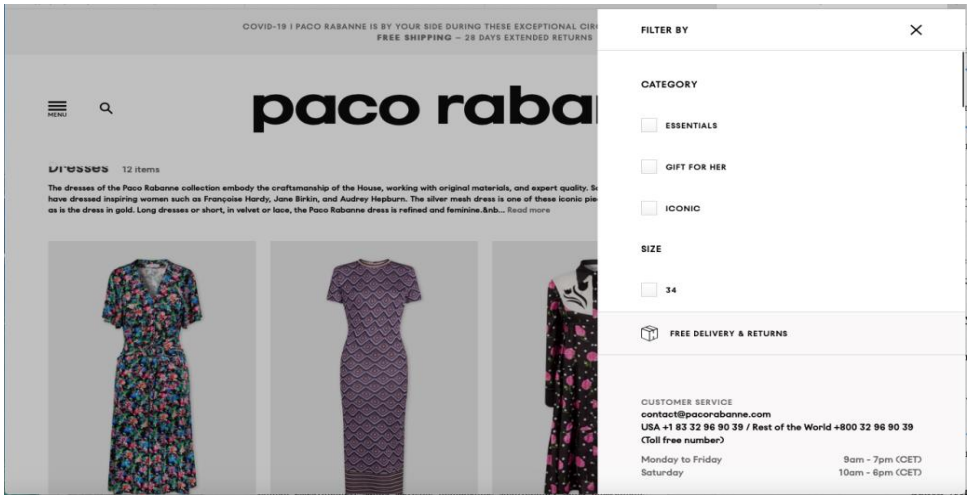

Рисунок 1.5 – Приклад інтерфейсу пошуку продукта в Hybris

Без перебільшень можна сказати, що Hybris-акселератор став революційним рішенням і багато в чому вирішив долю розвитку Java EE як технології, довівши її актуальність та незамінність на ринку розробкидля веб. “З коробки” представлені такі необхідні сервіси, як пошук товарів, додавання в кошик, оформлення замовлення, персоналізація акаунту користувача, а також широкий набір утиліт та кокпітів (cockpit, панелей управління) для адміністрування.

Платформа містить влику кількість високоефективних кокпітів. Вона має власну WCMS (Web Content Management System), гнучкий Backoffice-модуль для адміністрування, модуль для управління продуктами Product Cockpit, консоль адміністрування HAC та багато інших додатків для інтеграції з платіжними провайдерами, хмарними сховищами тощо. Однак серед наявних продуктів відсутні засоби оцінки ефективності і планування резервного копіювання даних.

1.6 Кластеризація у SAP Hybris

У комп’ютерних системах під кластером розуміють групу серверів (нод), які діють як єдина система й забезпечують високу доступність, оптимізоване балансування навантаження, а в деяких випадках і паралельне обчислення [11].

Розглянемо схему обробки даних у мультикластерному середовищі Hybris, наведену на рисунку 1.5.

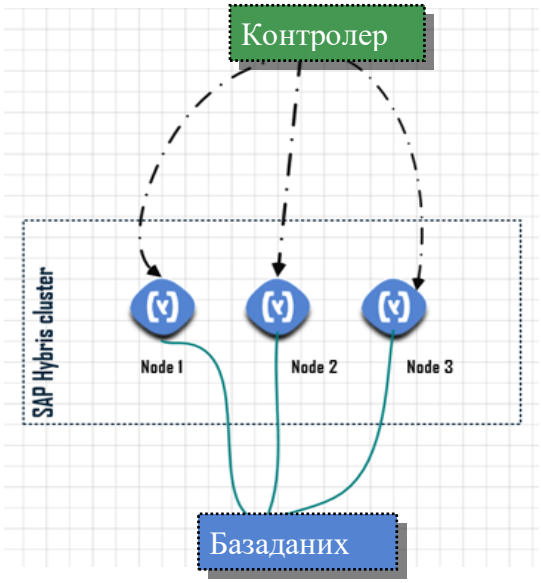

Рис.1.5 – Багатокластерне середовище в SAPHybris

У SAP Hybris кластери підтримуються ООТВ за допомогою кількох опцій на вибір. Найпростішим є використання інструментарію Jgroups: знадобиться лише кілька рядків .properties-файлу. Приклад коду для конфігурації:

```
# увімкнути режим кластера
clustermode = true
# обрати метод jgroups
cluster.broadcast.method = jgroups
```

Таким чином, кластеризація - потужний і простий механізм балансування навантаження та збільшення продуктивності роботи сервера в Hybris.

1.7 Постановка задачі

Здійснений аналіз предметної області дозволив конкретизувати мету роботи, яка полягає у визначенні найефективніших методів резервного копіювання даних у багатокластерному середовищі акселератора електронної

комерції Hybris, виявити переваги і недоліки цих методів та критерії ефективності з урахуванням потреб бізнесу. Акцент необхідно встановити на розробці гнучкого, ізольованого й безпечного досліджуваного середовища з урахуванням актуальних незалежних і залежних змінних-критеріїв, особливу увагу звернувши на показники RTO та RPO.

Для вирішення поставленої задачі необхідно:

– здійснити аналіз методів, моделей і програмних засобів резервного копіювання та післяаварійного відновлення даних;

– розробити методику і визначити критерії ефективності для резервного копіювання та післяаварійного відновлення даних у Hybris;

– виявити основні метрики, обрати аналізовані методи. Спроектувати сервіси, фасади та аспекти. Створити доменну модель рішення для платформи Hybris;

– розробити core-модуль з основною логікою програмної системи, web services-модуль з реалізацією кросплатформеного RESTful API та storefront-модуль з UI для управління процесами та графічного відображення результатів;

– розмістити готовий програмний продукт на ресурсахдля програм із відкритим сирцевим кодом (GitHub);

– провести оцінювання ефективності існуючих технологій за допомогою розробленого програмного продукту. Узагальнити отримані дані. Розробити рекомендації.

2 АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ РЕЗЕРВНОГО КОПІЮВАННЯ І ВІДНОВЛЕННЯ ДАНИХ У HYBRIS

У контексті даного дослідження альтернативами для резервного копіювання даних вважатимуться програмні засоби, сервіси чи утиліти для SAP Hybris Commerce, які дозволяють робити повний або частковий експорт даних локально або у хмарне сховище, з подальшим імпортом (післяаварійним відновленням) цих даних на сервері. Розглядатиметься також можливість проведення замірів показників ефективності наведених методів.

Пошук аналогів відбувався як усередині платформі Hybris (OOTB), так і серед зовнішніх ресурсів: спеціалізованої площини для завантаження додатків Hybris Marker place та джерел із відкритим сирцевим кодом.

2.1 Технологія імпорту-експорту ImpEx

ImpEx - скорочено “import-export” - уніфікований протокол створення (INSERT_UPDATE), оновлення (UPDATE), видалення (REMOVE) записів БД, а також експорту наявних рядків. Базується натехнології Flexible Search і представляє собою текстовий файл, сервіси розбору (інтерпретації), імпорту та експорту. Запит Flexible Search можна розглядати як посередник між об’єктною моделлю Hybris(NoSQL) та рядками у SQL базах даних. Він створюється у зручному для користувача форматі, незалежному від реляційної БД, згодом конвертується у SQL-запит відповідно до обраного типу СУБД.

Формат ImpEx дуже зручний для користувача, бо не вимагає знання деталей реалізаціїї в конкретній СУБД, підходить для створення великої кількості даних, легко читається, може бути імпортований автоматично, базується на CSV-форматі. Підтримуються макроси, функції, директиви.

ImpEx-скрипти поділяються на 2 типи: для імпорту та експорту. Особливість полягає у тому, що вони взаємоконвертуються: при експорті формується ImpEx-файл для імпорту обраних даних, а також додаткові CSV-файли, медіа-об’єкти тощо. При імпорті рядки заносяться у БД і згодом можуть бути експортовані. Приклад коду для імпорту каталог версій:

```
$productCatalog=electronicsProductCatalog
$catalogVersion=catalogversion(catalog(id[default=$productCatalog]),
version[default='Staged'])
$languages=ja,en,de,zh
INSERT_UPDATE Catalog;id[unique=true]
;$productCatalog
INSERT_UPDATE CatalogVersion;catalog(id[unique=true];version[unique=true];
active;languages(isoCode);readPrincipals(uid)
;$productCatalog;Staged;false;$languages;employeegroup
;$productCatalog;Online;true;$languages;employeegroup
```

Приклад коду для експорту каталог версій:

```
$productCatalog=electronicsProductCatalog
$version=Online
"#% impex.setTargetFile( ""Products.csv"" );"
INSERT_UPDATE CatalogVersion; catalog(id)[unique=true];
version[unique=true];
"#% impex.exportItems(""SELECT {CV:pk} {CatalogVersion as CV JOIN Catalog
as C ON {CV:catalog}={C:PK}} WHERE {C:id}='$productCatalog'
AND {CV:version}='$version'"");"
```

Як бачимо, синтаксис імпорту-експорту є легким для сприйняття за рахунок використання нуменклатури шару моделей: імена типу та полей у запитах співпадають з відповідними сутностями Java-класів, а не з реальними іменами таблиць у БД. Окрім того, підтримка макросів, директив і програмних вставок (“impex.exportItems” тощо) робить ImpEx дуже потужним.

Також технологія ImpEx пропонує користувачам інтеграцію з Hot Folder за допомогою Spring-конфігурації та підписки на події: при потраплянні до визначеної папки нових файлів у форматі CSV вони будуть конвертовані у ImpEx та автоматично імпортовані. Це значно прискорює імпорт нових об’єктів.

Імпорт-експорт базується на шарі сервісів: Import Service та Export Service із пакету de.hybris.platform.servicelayer.impex. Підтримує різні опції валідації (“суворий”, “розслаблений” тощо) і режим виконання коду. Бізнес-користувач може здійснити імпорт та експорт у панелі адміністрування HAC (Hybris Administration Console), як це показано на рисунку 2.1.

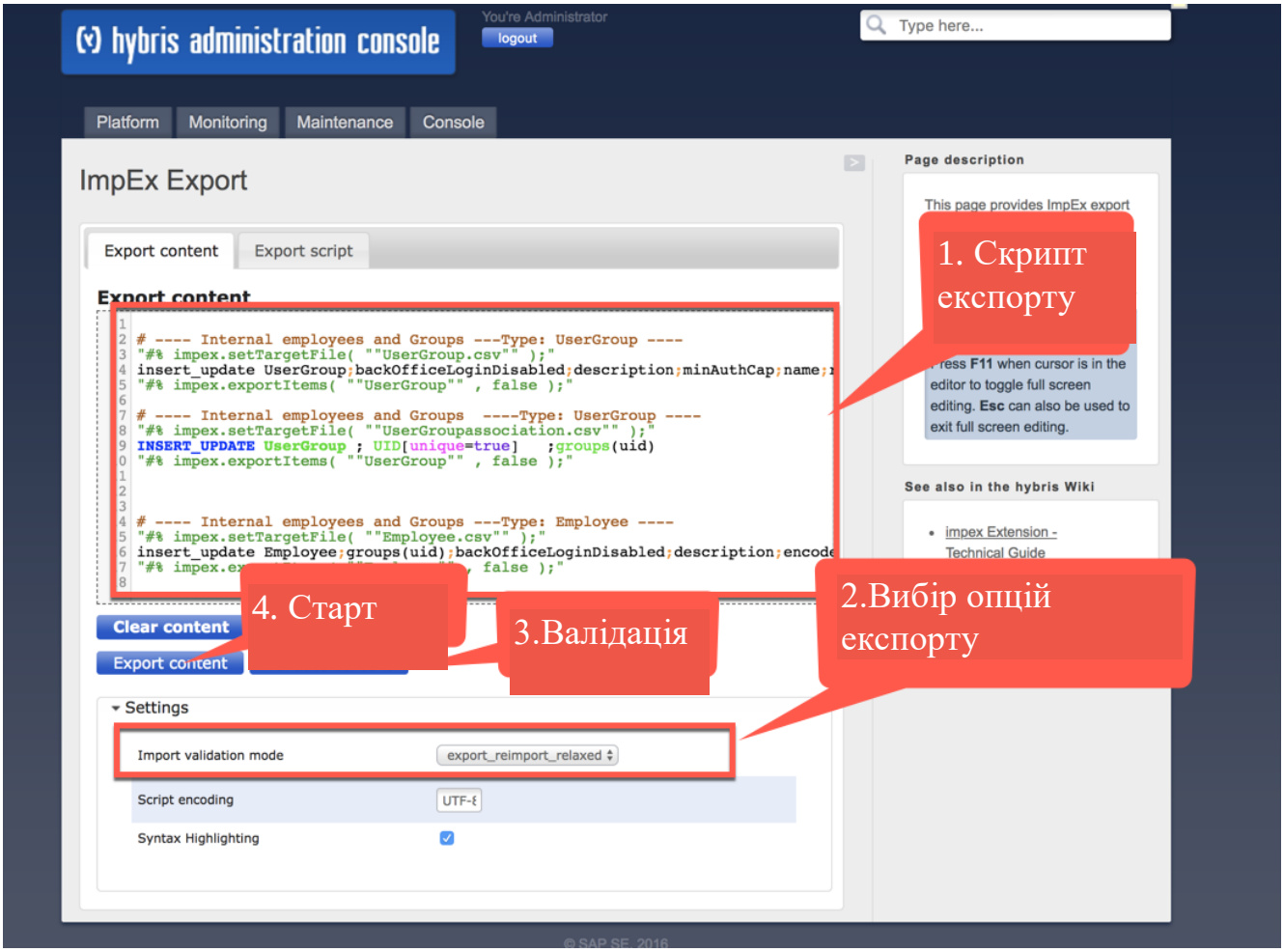

Рисунок 2.1 – Приклад імпорту у HAC

Таким чином, ImpEx - основний інструментарій для резервного копіювання ООТВ. Перевагами методу є:

– достатня швидкість виконання;

– функціонування на рівні сервісів, що забезпечує ізольованість;

– підтримка транзакцій;

– підходить як для імпорту, так і для експорту;

– універсальний формат, незалежність від СУБД;

– легкість у редагуванні й перегляді;

– може бути виконаний за допомогою HAC та безпосередньо з кодусервісів і фасадів.

Недоліки:

– необхідність вручну створювати файли для кожного окремого об’єкту (при імпорті) та для окремого типу сутності (при експорті):

– неможливість експортувати одразу весь вміст сайту чи каталогу;

– відсутні засоби оцінки ефективності імпорту й експорту, показника втрат, моніторингу основних метрик, статистики;

– медіа не включаються до експортованого архіву;

– резервні копії створюються лише локально.

2.2 Менеджер ImpExManager

Клас ImpExManager із пакету de.hybris.platform.impex.jalo є принципово відмінним методом імпорту та експорту в Hybris, який, так само як і попередній аналог, надається платформою “із коробки”. В той час, як попередня технологія базувалася на сервісному шарі, менеджер імпорту працює на Jalo-рівні.

У платформі Hybris Jalo - це функціональний рівень, який тісно сполучений із моделлю даних та бізнес-логікою. Jalo-класи генеруються на основі моделі даних. У нових версіях платформи (6.x, 18.x, 19.x)Jalo вважається застарілим і помічений анотацією @Deprecated [12]. Це означає, що платформа поки що підтримує його як спадковий код, однак використання таких класів є небажаним.

Основним недоліком Jalo є залежність від структури даних. Якщо модель даних змінюється, бізнес-логіка теж може потребувати адаптації. Для зменшеннярівня зчеплення між моделлю даних та логікою програми, рекомендується використовувати рівнь сервісів [13].

Клас ImpExManager містить такі методи для імпорту-експорту: importData; importDataLight, exportData, exportDataLight. Особливої уваги варті light-версії методів, оскільки, як заявлено у документації, вони забезпечують швидкість, значно вищу за відповідні методи на шару сервісів. Методи importDataLight та exportDataLight працюють із даними без використання кронджоб. Цим

обумовлюється їхня висока швидкість. При цьому кількість максимально дозволених проходів встановлюється як незкінченність.

Налаштування імпорту чи експорту не можуть бути сконфігурован через UI, як у випадку кронджоби. Результат виконання імпорту - екземпляр Jalo-класу Importer, експорту - об’єкт Exporter, який містить інформацію про здійснений процес і його статус. Слід звернути увагу, щологування до бази даних недоступне при використанні цих “полегшених” версій імпорту-експорту. Натомість можна використовувати стандарний log4ji з записом до файлу.

Перевагами обраного методу є:

– швидкість виконання;

– простота використання у коді.

Недоліки:

– відсутність кронджоб, які допомагають прослідкувати за статусом виконуваної операції;

– використання застарілого Jalo-рівня;

– відсутність транзацій;

– відсутність UI для конфігурації та метрик ефективності.

2.3 Дамп бази даних

Дамп БД - текстовий файл, що містить набір команд, які потрібно запустити на SQL-сервері, щоб створити базу даних із правильною внутрішньою структурою (схемою, таблицями, індексами) і заповнити її релевантними значеннями.

Переважно це тип резервного копіювання, який робиться на логічному рівні і може використовуватися для відновлення вмісту БД після часткової або повної втрати даних. На відміну від фізичної резервної копії бінарних файлів СУБД (які

використовуються виключно сервером і які неможливо скопіювати без його зупинки), процедура дампу може відбуватися при запущеному сервері.

У Hybris OOTB дамп підтримується для OracleSQL, MySQL, HSQL, PostgreSQL, SAPHANA та ін.. Поділяється на 2 типи: при ініціалізації системи та за вимогою.

У першому випадку файли дампу створюються автоматично тоді, коли відбувається процес ініціалізації, яка запускається командою:

```
cd hybris/bin/platform/ && ant initialize
```

При цьому повністю стираються попередні дані (включно з таблицями і ключами) зі схеми бази даних. Генеруютьсяокремі файли для:

– DDL (Data Definition Language statements) - SQL-команди для створення структури, тобто визначення нормалізованих таблиць із первинними та зовішніми ключами, а також індексами. Генеруються із XML-розмітки шару моделей, створеної в файлах *-items.xml.

– DML (Data Modeling Language statements) - SQL-команди для наповнення таблиць записами. Конвертуються з .impex-файлів, які імпортуються з initialdata-модулів.

Таким чином, дамп першого типу забезпечує бекап лише тих записів, які наявні у початковому стані системи, тобто присутні в initialdata-модулях. Це дані про країни, способи доставки, каталоги, сайти, магазини, групи користувачів. Взагалі кажучи, він не може бути розглянутим з метою післяаварійного відновлення, бо повністю знищує наявні у системі дані.

У свою чергу дамп за вимогою є методом повного резервного копіювання актуального стану системи і може включати медіа-об’єкти. Бекап бази даних запускається за допомогою інтерфейсів, надані засобами СУБД: графічних чи з командної строки.

Для запуску процедури дампу з командного рядка в MySQL використовується команда:

```
mysql script : mysqldump -uuser -ppassword nom_base > path_to_dump_date.sql
```

Приклад створення дампу БД із використанням графічного інтерфейсу db Forge Studio для MySQL показаний на рисунку 2.2.

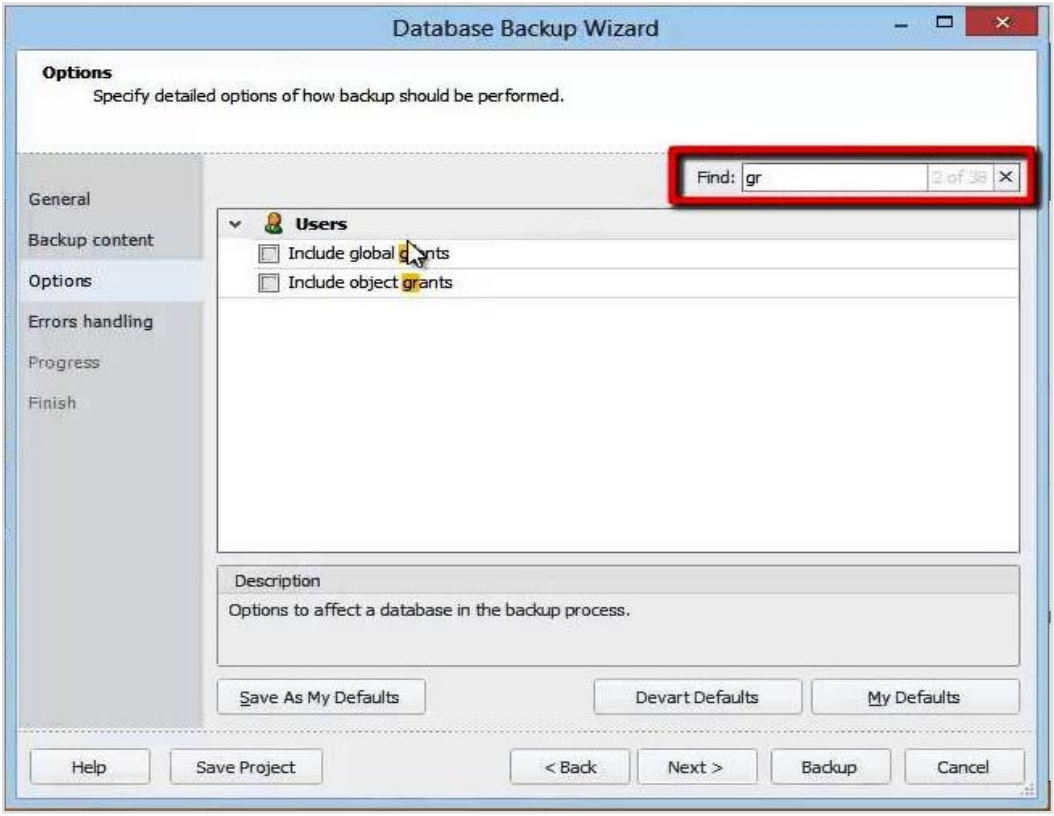

Рисунок 2.2 – Створення дампу бази даних для MySQL

Рекомендується в імені файлу вказувати дату бекапу. Варто зазначити, що експортовані медіа-об’єкти містять посилання на реальні фізичні файли у системі. При пошкодженні файлів відповідні рядки дампу не будуть імпортовані успішно. Враховуючи це, обов’язковою скаладовою є архівація медіа-директорій:

```
linux script zip -r repertoire path_to_zip.zip
```

Де “path/to/dir/date.zip” - шлях до нового архіву. Після аварії потрібно розархівувати zip з медіа-файлами, після чого відновити базу даних:

```
restore data base mysql script : mysql -uusername -ppass-Dnom_base < path_to_dump_date.sql
```

Дамп БД може бути використаний з метою:

– бекапу й відновлення даних під час потенційної аварії;

– реплікації, щоб сконфігурувати допоміжні slave-сервери й покращити продуктивність і надійсність застосунку;

– міграції до інших (аналогічних) СУБД.

До переваг методу слід віднести:

– швидкість виконання значно вища за ImpEx, оскільки читання з БД відбувається напряму й відсутня додаткова конвертація з SQL;

– файл дампу можна використати в інших системах без платформи Hybris;

– допускається створення дампу за вимогою як через командний рядок, так і з графічним інтерфейсом (MySQL Workbench, dbForge Studio тощо);

– підтримка бекапумедіа-об’єктів.

Недоліки:

– СУБД-залежність: може бути використаний тільки в рамках аналогічної системи управління базами даних;

– складність резервного копіюваннямедіа;

– відсутність транзакцій;

– неможливість моніторингу в рамках платформи Hybris;

– потребує додаткових програмних засобів для імпорту з файлу дампу;

– оцінка швидкості та продуктивності бекапу може бути проведена тількизасобами додаткового програмного забезпечення;

– не підходить для HSQL.

2.4 Розширення Aimprosoft Hybris Commerce CBM

Commerce Cloud Backup and Migration (Commerce CBM) – це розширення для SAP Commerce платформи версії 1808/1811 (з B2C Accelerator).Забезпечує програмне рішення для створення та управління бекапами та сховищами медіа-об’єктів на різних хмарних провайдерах [14].

Розширення має зручний інтерфейс, оформлений як система віджетів у перспективі Hybris Backoffice. Зручною є функція перегляду історії задач імпорту та експорту одразу післяїх виконання, як показано на рисунку 2.3.

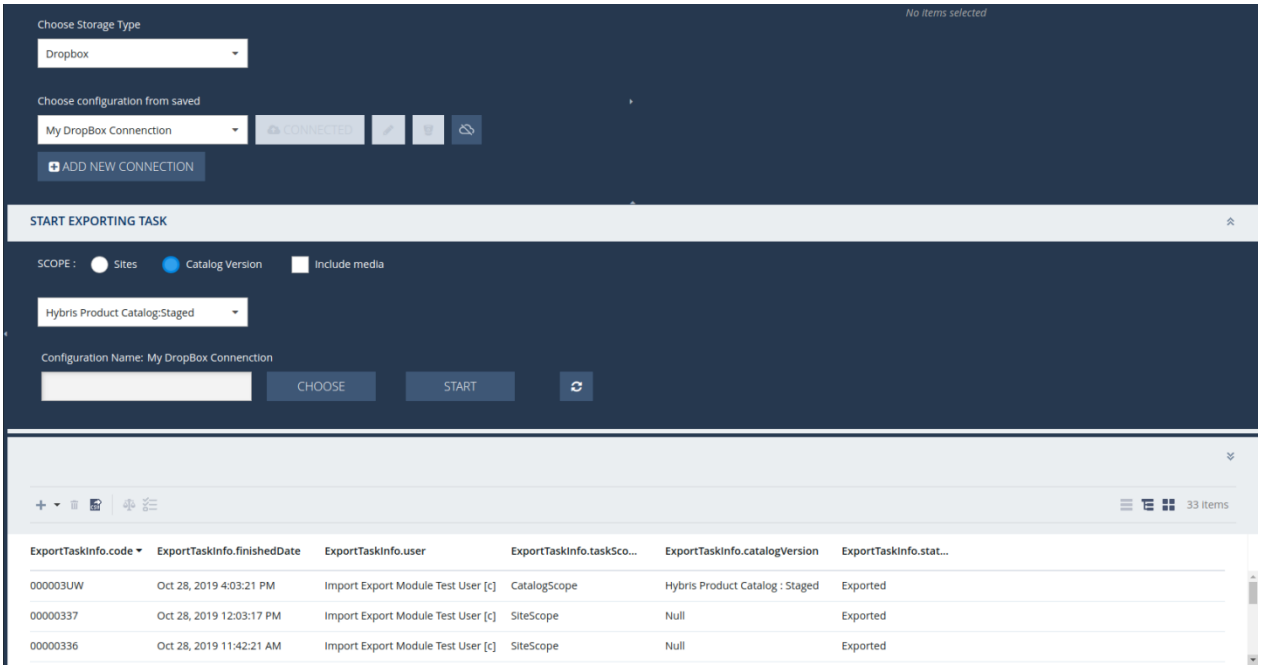

Рисунок 2.3 – Графічний інтерфейс CommerceCBM

SAP Commerce CBMv.1.0 дозволяє користувачам підключити необмежену кількість Amazon S3 та Dropbox акаунтів [14]. Це робить післяаварійне відновлення більш гнучким, оскільки дозволяє записувати файли бекапів й імпотрувати безпосередньо з хмари. Приклад підключення бакету із використанням конфігурацій доступу Amazon S3 показаний на рисунку 2.4.

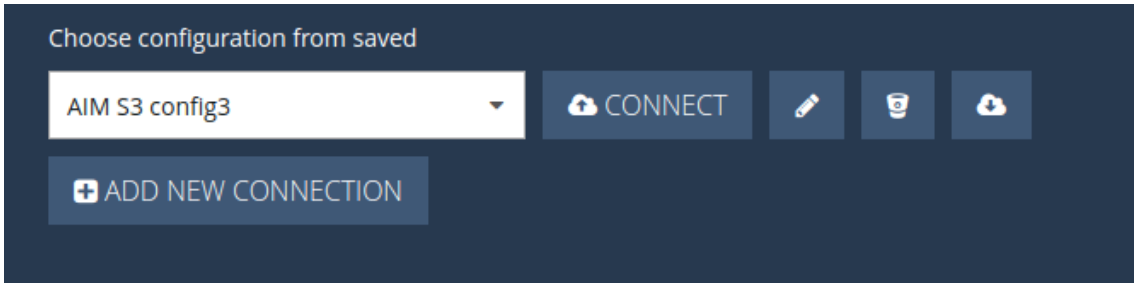

Рисунок 2.4 – Підключення акаунту AmazonS3 в Commerce CBM

Розширення доступне для безкоштовного завантаження на SAP App Center, сторінка інсталяції показана на рисунку 2.5.

Рисунок 2.5 – Сторінка завантаження розширення на SAP App Center

Переваги CBM:

– створення бекапів сайтів і каталогів з усім медіа контентом “в один клік”;

– зберігання й імпорт бекапів (ImpEx, CSV, медіа-файлів) в одному zip-архіві;

– автоматичне розпізнавання типів, необхідних для включення в експорт, залежно від обраного рівня бекапу (тільки контент, певний каталог, весь сайт);

– швидке відновлення з бекапів на будь-якому сервері;

– дозволяє зберігати копії сайтів для різних команд розробників;

– СУБД-незалежне рішення;

– нявність історії виконаних експорт- та імпорт- завдань.

– підтримка транзакцій.

Недоліки:

– швидкість виконання нижча за ООТВ-аналоги;

– потребує створення додаткового додатку у Dropbox;

– відсутні метрики швидкості, % виконання, завантаженості процесорів.

Підсумовуючи результати аналізу аналогів, варто зазначити відсутність у всіх наведених альтернатив засобів вимірювання ефективності резервного копіювання та післяаварійного відновлення з точки зору DRaaS.

3 КРИТЕРІЇ, МЕТОДИКА ТА РЕЗУЛЬТАТИ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МЕТОДІВ РЕЗЕРВНОГО КОПІЮВАННЯ ТА ПІСЛЯАВАРІЙНОГО ВІДНОВЛЕННЯ

3.1 Побудова моделі рішення

Під моделлю у контексті дослідження ефективності резервного копіювання та післяаварійного відновлення у середовищі Hybris будемо вважати:

– вимірюваний у часі, автоматичний, із заданим розподілом і густиною процес створення тестових об’єктів шару моделі (Model Data Layer), який імітує реальне наповнення серверних даних під час пікових та стандартних навантажень;

– періодичний, із фіксованою частотою запуск експорту даних із використанням аналізованого методу - модель резервного копіювання на сервері; періодичний, із заданою частотою процес часткового або повного пошкодження даних - прототип аварії на сервері;

– запуск імпорту із наявних резервних копій, створених аналізованим методом - модель післяаварійного відновлення.

Виділимо основні припущеннята обмеження в моделі:

– для створення об’єктів використовуються лише основні сутностідоменної моделі Hybris: Користувач (UserModel), Продукт (ProductModel) та Замовлення (OrderModel);

– розподіл між типами сутностей відбувається за сталим коефіцієнтом;

– частота експорту даних еквівалентна RPO;

– швидкість післяаварійного відновлення еквівалентна RTO;

– післяаварійне відновлення починається одазу після аварії, часом на запуск процесу бізнес-користувачем можна знехтувати;

– аналізованими методами є наведені у попередньому розділі найкращі альтернативи: імпорт-експорт ImpEx (OOTB), менеджер імпорту ImpEx Manager (OOTB), дамп бази даних (стороннє програмне забезпечення) та розширення Hybris Aimprosoft CBM (стороннє програмне забезпечення).

– одночасно використовується лише один із аналізованих методів резервного копіювання.

3.2 Критерії ефективності резервного копіювання

Виділимо аналізовані критерії, представлені як незалежні й залежні змінні. Оскільки основною метою післяаварійного відновлення як сервісу є зменшення показника RTO, ця метрика є базовою залежною змінною. У той же час RPO, який обирається під час планування DR - основна незалежна змінна.

Незалежні критерії $x_i, x \in (1, n)$ повинні бути задані як вхідні парамери обчислення. Повний перелік незалежних мінних наведений у таблиці 3.1, де вказаний їх пріорітет за шкалою від 1 до n, деn- кількість критеріїв, вважаючи, що 1 - максимальний бал.

Таблиця 3.1 – Незалежні критеріїx_i

№	Назва критерію	Тип змінної	Пріоритет
1	Об’єктивна точка відновлення RPO, год	Числова (Double)	1
2	Обсяг даних	Числова (Integer)	2
3	Кількість потоків	Числова (Integer)	3
4	Частота процесора	Числова (Double)	4
5	Обсяг RAM процесора	Числова (Double)	5
6	Наявність транзакцій	Булева (Boolean)	6

Залежні критерії $y_i, y \in (1, n)$ є результатами обчислення й дозволяють виявити фукцію залежності результату (ефективності) роботи методів резервного копіювання та післяаварійного відновлення від обраних змінних x_i.

Пріоритезація здійснювалася з урахуванням критерів, найбільш цінних для власників бізнесу. Слід зазначити, що рівень розвитку апаратного забезпечення дозволяє збільшувати потужність серверів за відносно невеликі кошти. Так, оренда надсучасного сервера DS-256 із процесором 2.9 ГГц Turbo, обсягом оперативної пам’яті 256 ГБ та 2 ТБ дискового простору на швидкісних SSD-накопичувачах обійдеться власникові бізнесу лише в 400 доларів США [15]. У той час як втрата інформації оцінюється в десятки тисяч доларів.

Тому критерії, пов’язані з даними, цінніші за ті, що пов’язані із апаратними характеристиками. Перелік відсортованих у порядку пріорітетності залежних змінних наведений у таблиці 3.2.

Таблиця 3.2 – Залежні критерії y_i

№	Назва критерію	Тип змінної	Пріоритет	Спосіб вимірювання
1	Об’єктивний час відновлення RTO, год	Числова (Double)	1	Aspect
2	Відсоток втрати даних	Числова (Double)	2	Aspect
3	Відсоток завантаженості процесора	Числова (Double)	3	OOTB
4	Кількість звернень до БД	Числова (Integer)	4	OOTB

Сформулюємо вимоги до подання й інтерпретації наведених критеріїв:

– залежні змінні повинні конфігуруватися бізнес-користувачем за допомогою графічного UI або шляхом вибору серверів з різними технічними характеристиками;

– незалежні змінні повинні обчислюватися й бути доступними для експорту за допомогою RESTful APi;

– результати заміру метрик повинні бути збережені у базі даних;

– моделі аудиту повинні містити вичерпну інформацію про стан системи до та після “аварії” - порушення цілісності даних;

– наведені критерії повинні вимірюватися лише для тестового ізольованого каталогу, реальні дані кластеру не повинні взаємодіяти із системою.

3.3 Методика оцінювання ефективності

Залежні критерії y_i , спосіб вимірювання яких позначений як ООТВ (відсоток завантаженості процесора, кількість звернень до бази даних), можуть бути виміряні шляхом сервісів, наданих платформою Hybris “із коробки”.

Об’єктивний час відновлення RTO доцільно вимірювати за допомогою АОР-підходу, створивши відповідні аспекти для кожного аналізованого методу з поінткатами, анотованими як @Withing, в яких встановлюється початковий та кінцевий час виконання методів. Аспектно-орієнтоване програмування в Hybris підтримується бібліотекою AspectJ та фреймворком Spring. Воно є потужним та економним з точки зору ресурсів сервера механізмом “вклинення” у виконання методів на етапі компіляції або в рантаймі.

Розглянемо приклад аспекту:

```
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class SystemArchitecture {
  @Withing(“execution(ua.nure.nikonova.standbycore.importData..*)”)
    public void inWebLayer() {
   //логіка заміру показників часу виконання методу importData
}
}
```

Для обчисленя відсотку втрати даних при післяаварійному імпорті з бекапносія необхідно реалізувати сервіс та API з функцією експорту тестових даних, і здійснювати відповідний HTTP-запит до та після “аварії”. Отриману відповідь у

форматі JSON необхідно десеріалізувати та порівняти цілісність експортованих об’єктів.

Для оцінки цілісності даних може бути використаний хеш-код об’єктів, який повинен включати в себе усі значимі поля відповідної моделі, окрім PK із класу GenericItemModel (батьківського класу усіх об’єктів-моделей у Hybris, аналог Object у Java). PK - унікальний ідентифікатор усіх моделей, який створюється при записі моделі до бази даних, генерується автоматично й не гарантує еквівалентності при повторному імпорті того ж об’єкту. Окрім PK, моделі мають інші унікальні ключі, комбінація яких повністю ідентифікує модель, і зберігається при повторному занесенні аналогічної моделі у базу даних. Саме вони повинні бути прийняті до уваги при генерації хеш-коду. Наприклад, код та версія каталогу для продуктів. Об’єкти відношень однозначно описуються за допомогою джерела (source) та цілі (target).

3.4 Результати оцінювання ефективності методів резервного копіювання та післяаварійного відновлення даних

Оскільки логіка програмної системи передбачає велику кількість змінних і ступенів свободи, виключимо Парето-неоптимальні альтернативи із множини аналізованих методів [16].

Так як значення кількісних критеріїв на даному етапі невідомі, сформуємо набір основних якісних критеріїв, розглянутих у попередніх розділах дослідження: СУБД-незалежність, максимальний рівень експорту, тип UI, простота установки, підтримка хмарного середовища.

Виділимо шкали за критеріями:

– СУБД-незалежність: присутня, відсутня;

– максимальний рівень експорту: кластер, сайт, каталог, окремі об’єкти;

– тип UI: відсутній, HAC (Hybris Administration Console), Desktop, термінал, Backoffice;

– простота установки: ООТВ, стороннє програмне забезпечення, розширення.

– підтримка хмарного середовища: присутня, відсутня.

Результат векторної оцінки за критеріями наведений у таблиці 3.3.

Таблиця 3.3 – Векторна оцінка за якісними критеріями

№		СУБД-незалежність	макс. рівень експорту	тип UI	Простота установки	Підтримка хмарного серед.
1	Технологія ImpEx	Присутня	Каталог	HAC	ООТВ	Відсутня
2	ImpEx Manager	Присутня	Окремі об’єкти	Відсутній	ООТВ	Відсутня
3	Дамп бази даних	Відсутня	Кластер	Desktop, термінал	Стороннє ПЗ	Відсутня
4	Commerce CBM	Присутня	Сайт	Backoffice	Розширення	Присутня

Нормалізувавши оцінки за критеріями, отримали 2 Парето-неоптимальні альтернативи: ImpEx Manager та Дамп бази даних, які можна вилучити із подальшого дослідження.

Складемо план тестування для порівняння технології ImpEx та розширення Commerce CBM, а також оцінки впливу незалежних критеріїв x_i, виділених у розділі 3, на показники y_i. Сценарії, пов’язані із даними, що виконуються для обох альтернатив з усіма доступними значеннями RPO і сталим коефіцієнтом hourlyAmount:

– сценарій 1: час виконання методів (RTO) у залежності від обраного RPO: 1, 2, 4, 8, 12 та 24 години;

– сценарій 2: відсоток завантаженості процесора в залежності від RPO;

– сценарій 3: кількість звернень до БД в залежності від RPO. Сценарії, що пов’язані із потужністю сервера, виконуються для 1 альтернативизі сталим RPO;

– сценарій 4: RTO в залежності від обсягу оперативної пам’яті RAMта потужності процесора CPU;

– сценарій 5: RTO в залежності від наявності транзакцій.

Таким чином, розглянуті усі найважливіші фактори впливу, суттєві при плануванні післяаварійного відновлення в Hybris. Дослідження проводяться для СУБД HSQL (HyperSQL). За час роботи RTO прийнято час імпорту даних зі створеного заданою альтернативою бекап-досія при експорті.

Сценарій 1. На рисунку 3.1 зображені середні значення 10 запитів для кожної комбінації RTOдля альтернатив: технологія ImpEx та розширення Commerce CBM.

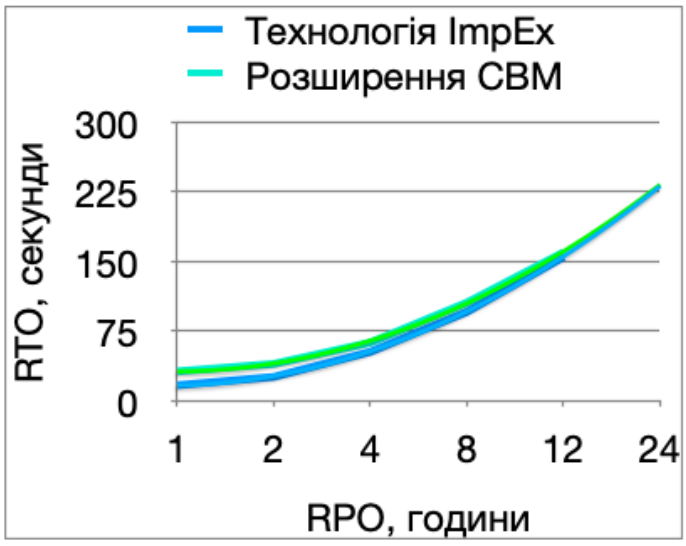

Рисунок 3.1 - Залежність RTO від RPO

Як бачимо з графіка, при невеликих значеннях RPO, коли генерація скриптів та створення додаткових сутностей відбувається часто, доцільнішим є використання методики “із коробки” - технології ImpEx на згенерованих вручну скриптах експорту. Водночас, коли RPO набуває показників, більших за 4 години,

ефективність СВМ майже досягають аналогічних позначок для базової технології ImpEx.

Сценарій 2. Дослідження проводилися на основі Intel i5 4x2.9ГГц. Середній результат для 10 значень кожної комбінації RPO показаний на рисунку 3.2.

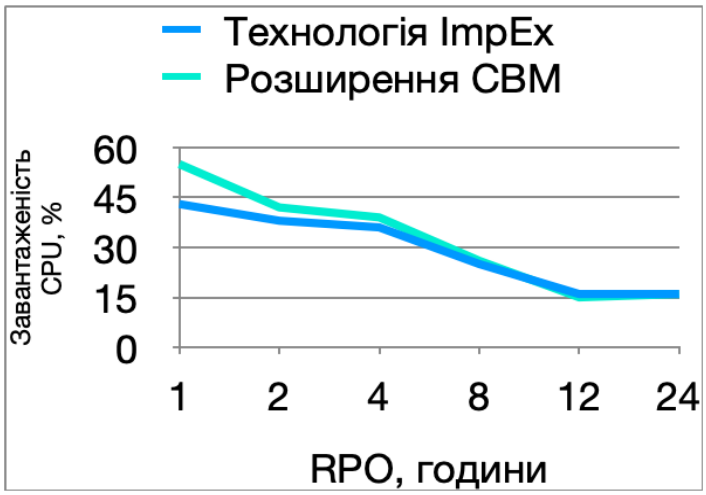

Рисунок 3.2 – Залежність завантаженості CPUвід RPO

Сценарій 3. Кількість звернень до БД зазначене з урахуванням усіх процесів у платформі. Результат серії експериментів наведений на рисунку 3.3.

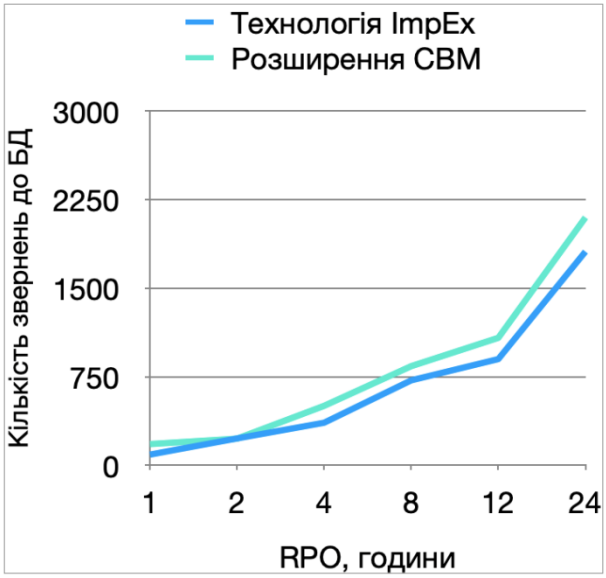

Рисунок 3.3 – Залежністькількості звернень до БДвід RPO

Подальші дослідження проводилися на базі таких локальних серверів:

– Intel i5 4x2.9ГГцCPU, 32 ГбRAM, 128 ГбSSD, Ubuntu 16.04 LTS;

– Intel i5 4x1.8 ГГц CPU, 8 ГбRAM, 128 Гб SSD, MacOS Catalina 10.15.3.

Сценарій 4. Використане значення RPO = 4 години, кількість даних = 400, метод – технологія ImpEx, досліджуваний носій RAM №1 = 32 Гб, RAM №2 = 8 Гб, CPU №1 = Intel i5 4x2.9 ГГц, CPU №2 = Intel i5 4x1.8 ГГц.

Кількість запитів - 10 для кожного сервера. Результати експерименту наведені на рисунку 3.4.

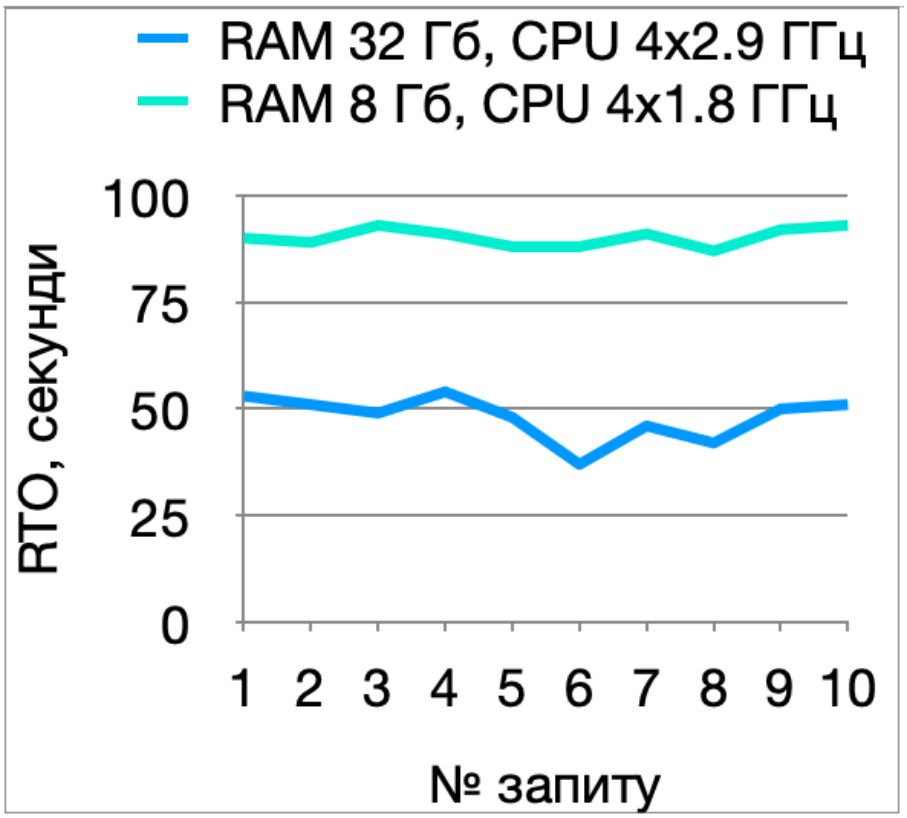

Рисунок 3.4 – Залежність RTO відпотужності сервера

Сценарій 5. Досліджуваний CPU= Intel i5 4x2.9 ГГц, обсяг RAM = 32 Гб, RTO = 4 години, метод резервного копіювання - технологія імпорту-експорту ImpEx. Кількість запитів - 10 для кожної опції (з транзакціями та без). Результати наведені на рисунку 3.5.

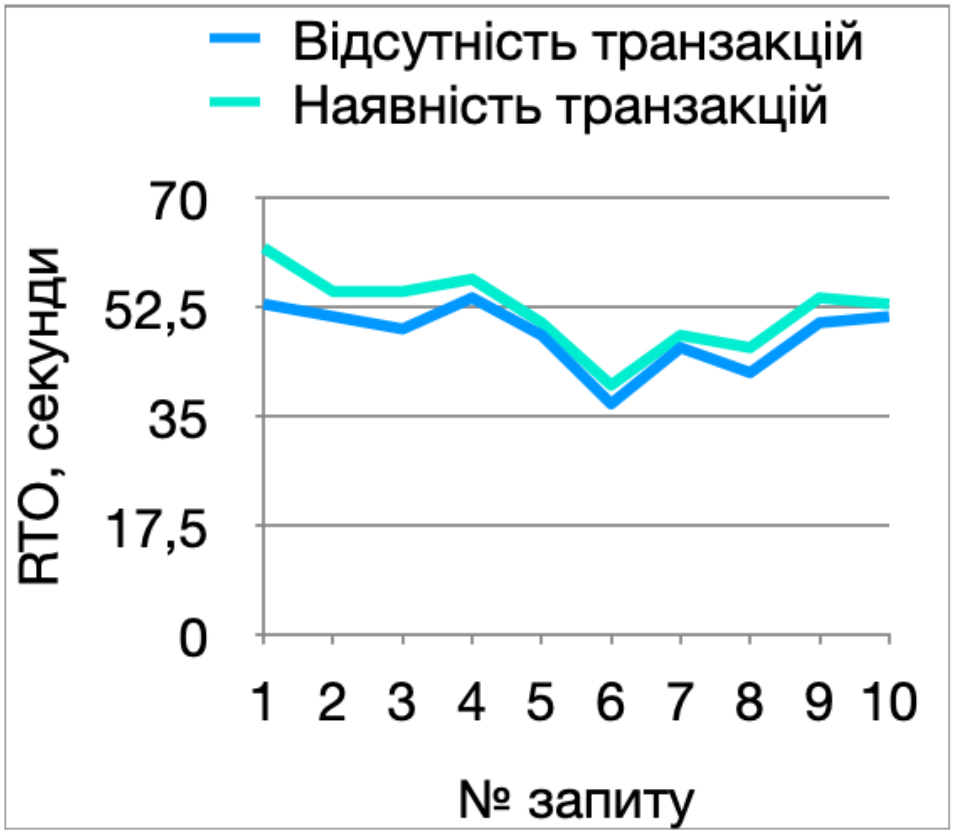

Рисунок 3.5 – Залежність RTOвід наявності транзакцій

У ході експериментів доведена гіпотеза про балансування між вартістю втрати даних та вартістю створення бекапів. Так, зі збільшенням показника RPOсуттєво збільшується час RTO, а отже і вартість втрачених даних стрімко зростає. Водночас збільшення часового проміжку RTO призводить до зниження навантаження на серверні ресурси і вартості створення бекапів.

Для пошуку оптимальної точки балансу між вартістю резервного копіювання та потенційними втратами при пошкодженні даних були нормалізовані одержані графіки. Таким чином одержали результат RPO, наближено рівний восьми годинам. Це означає, що при заданій конфігурації сервера та коефіцієнтові щогодинного створення нових даних найоптимальнішим варіантом є здійснення резервних копій один раз у вісім годин (три рази на добу).

Принцип пошуку оптимального рішення, позначеного як точка А перетину двох графіків, показаний на рисунку 3.6.

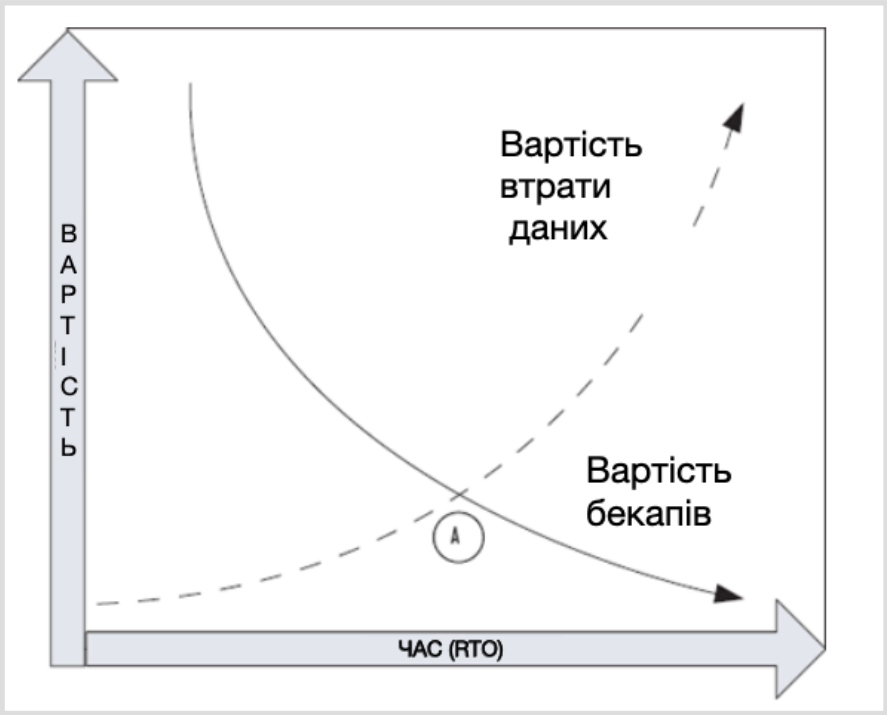

Рисунок 3.6 – Балансування між вартістю втрати даних та вартістю бекапів

Експерментально була визначена ступінь залежності об’єктивного часу відновлення RTO від об’єктивної точки відновлення RPO, процент завантаженості процесора від RPO для технології імпорту-експорту ImpEx та розширення Commerce CBM. Виявлено, що для низьких значень RPO (до 4 годин, тобто досить частого резервного копіювання) рекомендованим є використання технології ImpEx. Для більших RPO (понад 4 години) у CommerceCBM показники ефективності майже рівні аналогічним показникам для технології ImpEx. Однак додатково з’являється можливість прямого запису бекапів у хмарне середовище, а також створення історії завдань експорту та імпорту. Саме тому Commerce CBM рекомендується для використання у кластерах, де RPO перевищує чотири години.

Також рекомендується використання транзакцій, оскільки вони незначним чином збільшують показник RTO.

4 ОПИС ПРОГРАМНОЇ СИСТЕМИ ДЛЯ ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ МЕТОДІВ РЕЗЕРВНОГО КОПІЮВАННЯ ТА ПІСЛЯАВАРІЙНОГО ВІДНОВЛЕННЯ ДАНИХ У HYBRIS

4.1 Архітектура програмної системи

Для дослідження ефективності обраних методів розроблено набір розширень Hybris,об’єднаних у модульіз назвою Standby (дослівно перекладається як “опора” іє поширеним терміном в DRaaS). Розширення мають спільний простір імен (namespace) і поєднані за допомогою механізму залежностей (required extensions). Цей підхід повністю відповідає загальному архітектурному концепту платформи SAP Hybris Commerce. Структура проєкту в IntelJ IDEA показана на рисунку 4.1.

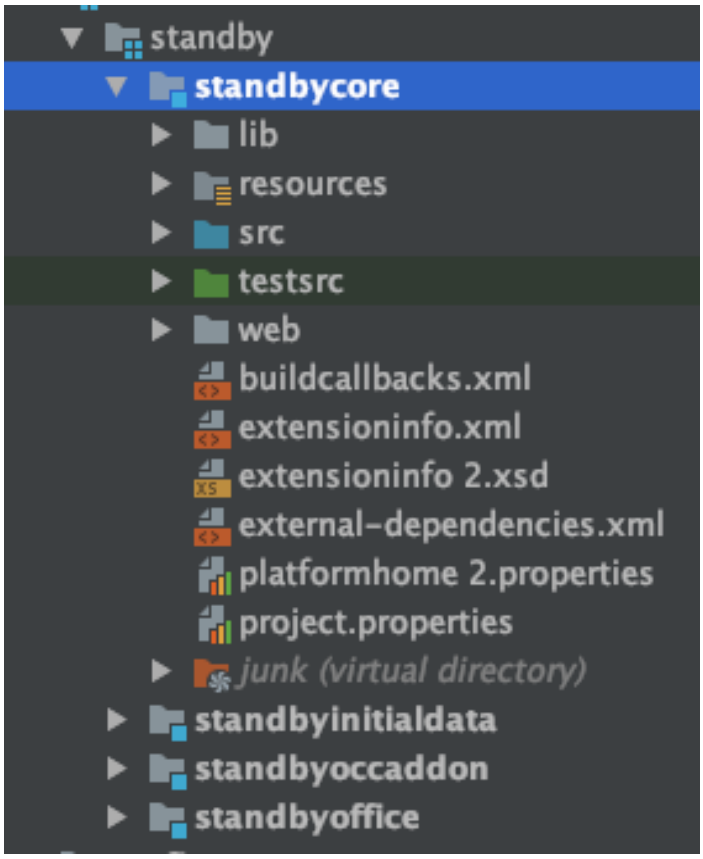

Рисунок 4.1 – Структура проєкту - модуль Standby

До модулю Standby увійшли такі розширення:

– Standbycore - основне ядро з логікою сервісів, аспектів, утиліт, а також моделями, необхідними для функціонування модуля;

– Standbyinitialdata - розширення, яке містить ресурси з ImpEx-файлами, необхідними для імпорту початкових даних (каталогів, каталог версій, кронджоб), які використовуються тестовою системою;

– Standbyoccaddon - аддон, який реалізує RESTful API з основними ендпоінтами, які надає модуль для гнучкої взаємодії;

– Standbyoffice - розширення з віджетами для панелі управління Backoffice, який надає зручний UIдля конфігурації та перегляду результатів.

Модуль Standby розгортається у платформі SAP Hybris Commerce версії 6.7 на базі серверного контейнера застосунків Apache Tomcat. Діаграма розгортання програмної системи показана на рисунку 4.2.

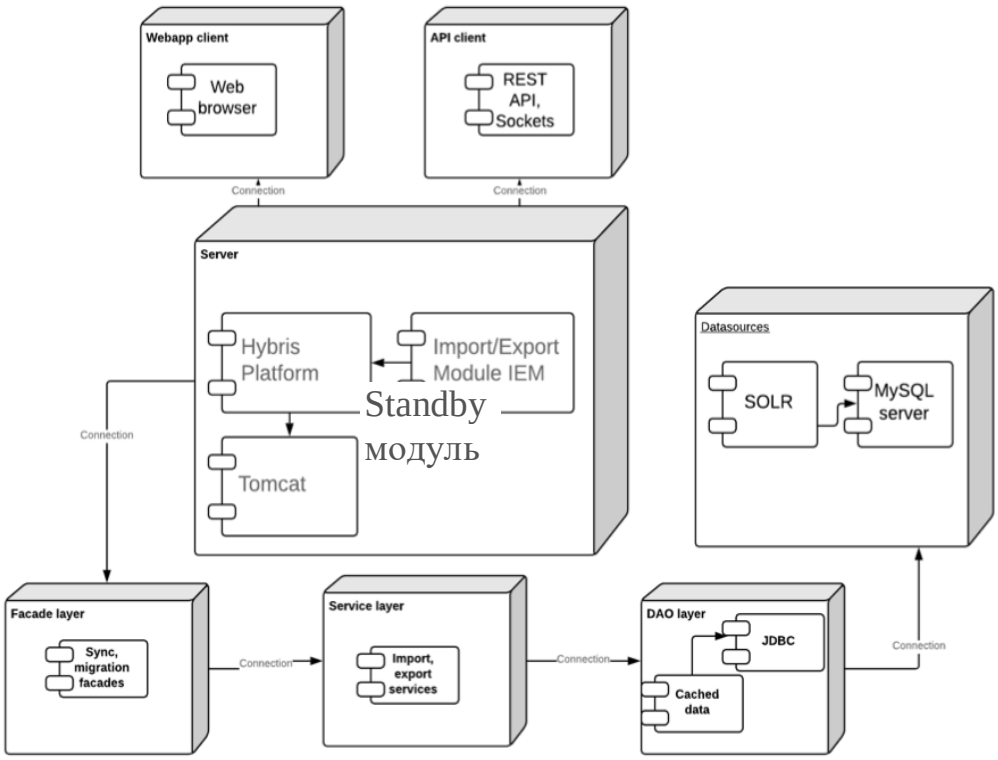

Рисунок 4.2 – Діаграма розгортання програмної системи Standby

Основна логіка проєкту зосереджена у розширенні stadbycore. Використовуються 3 основні пакети: job, aspect, service. Пакет job містить реалізацію Abstract Job Performable для задач генерації даних та порушення їхньої цілісності. Він використовує пакет service - сервіси для роботи з об’єктами доменної моделі - замовленнями (OrderModel), їхньої вибірки із БД та функції отримання Sha1-коду для стану об’єкту. Наведемо діаграму класів ядра standbycore на рисунку 4.3.

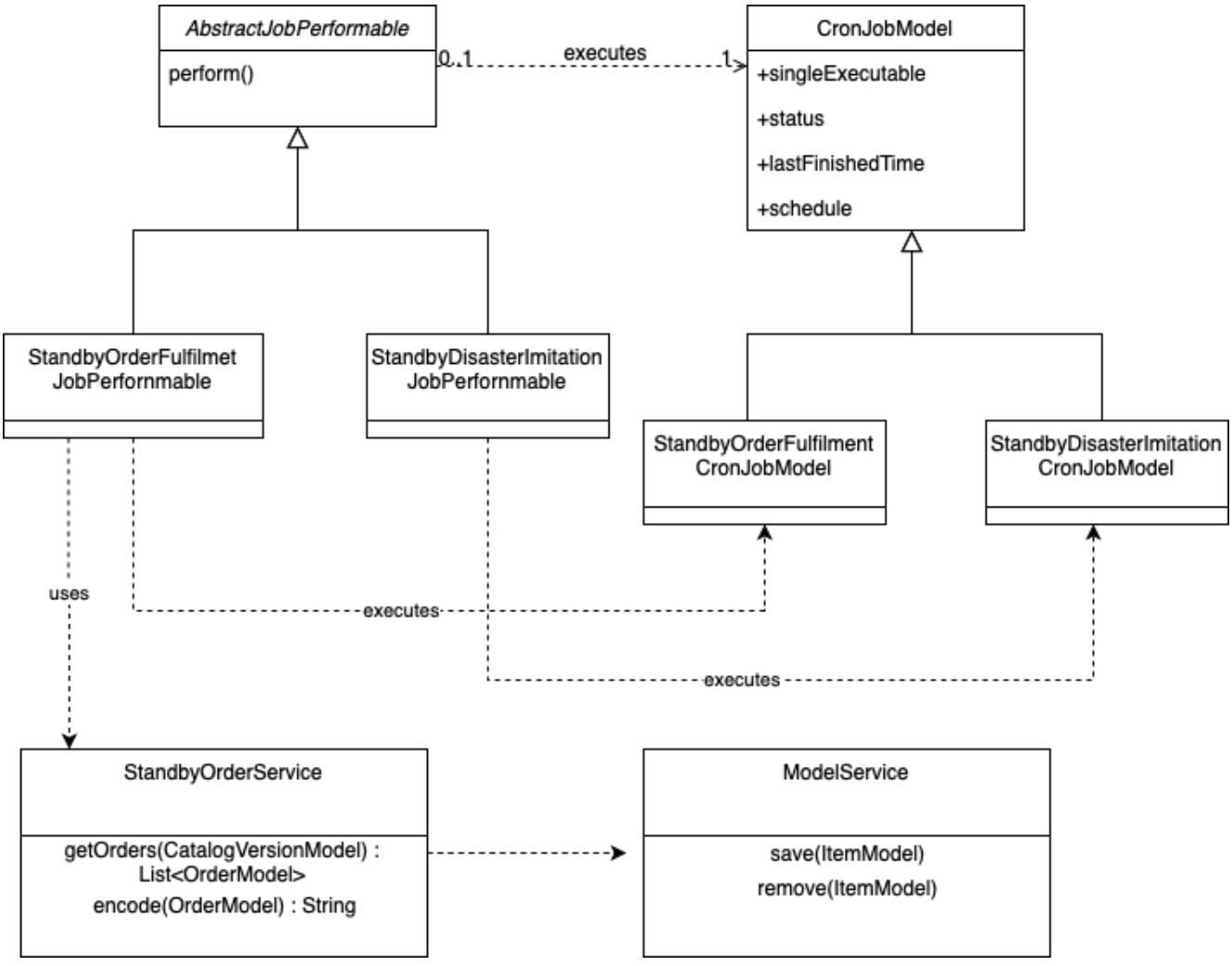

Рисунок 4.3 – Діаграма класів розширення Standbycore

Пакет aspect містить класи аспектів для вимірювання завантаженості CPU, відстока втрат та часу виконання методів. Вони створюють і зберігають модель

аудиту StandbyAudit, у якій містяться результати експериментів. Список StandbyAudit може бути отриманий за допомогою RESTful API запиту.

4.2 Опис обраних технологій

Основною мовою розробки обрана Java, оскільки вона використовується ядром та всіма розширеннями платформи SAP Hybris Commerce. Hybris сумісний із Java Development Kit (JDK) версії 1.7+ (яка відповідає JavaSE 7), але оскільки у більшості проєктів використовується версія JDK 1.8, для програмної системи Standby була обрана саме вона. Починаючи з версії 8, у JavaSE з'явилася підтримка лямбда-виразів, функціонального програмування, технологій Optional, StreamAPI, а також принципово нового, потужного DateTime API. Зазначимо, що характерною особливістю JDK є абсолютна зворотня підтримка, тобто, наприклад, у версії 1.8 повністю компілюється код з усіх попередніх версій. Це дуже важливо з точки зору бізнесу, який таким чином може підтримувати legacy-код.

У релізі JavaSE 8 відбулися значні удосконалення завдяки застосуванню лямбда-виразів - нового мовного засобу, який принципово змінює в кращу сторону підхід до програмування, додаючи можливості функціонального програмування. Лямбди здатні спростити і скоротити обсяг сирцевого коду для створення багатьох конструкцій, включно з анонімними класами. Вони вводять у синтаксис новий оператор і допомагають зберегти жвавість та проворність мови Java. З точки зору продуктивності, дуже важливим оновленням є новий програмний інтерфейс API потоків введення-виводу з пакету java.util.stream. У цьому оптимізованому пакеті підтримуються конвеєрні операції з даними. Також потужним інструментом удосконалення програмного коду є поява default-методів із реалізацією за замовчуванням в інтерфейсах. Таким чином, новий метод може бути введений в інтерфейс, не порушуючи його семантику і структуру, які вже

існують. Це захищає класи від надмірного використання наслідування абстракних класів і сприяє переходу до більш гнучких інтерфейсів.

Слід виділити такі особливості мови Java:

– простота;

– високий рівень безпеки;

– об’єктно-орієнтований підхід;

– підтримка аспектно-орієнтованого та функціонального програмування;

– надійність;

– багатопотоковість;

– кросплатформенність;

– висока ефективність;

– динамічність [17].

Для розробки модуля використаний фреймворк Spring MVC. Spring MVC - це надійний, гнучкий і добре спроектований фреймворк для швидкої розробки веб-застосунку на основі паттерну MVC (Model-View-Controller). Переваги використання цього фреймворку схожі на ті позитивні сторони, які дає використання бібліотек цієї компаніїу цілому. Більшість классів Spring розроблені як Java-біни, це дозволяє легко вставляти тестові дані [18].

Абстрактний принцип інверсії управління (Inversion of Control, IoC) реалізований у Spring за допомогою підходу Dependency Injection (DI). IoC-контейнер забезпечує важливий функціонал введення залежностей. Більшість бінів зберігаються як сінґлтони. Це допомагає розробникам вставляти залежності, такі як бізнес-сервіси, в рантаймі, рятує від дублювання коду.

Spring фреймворк підтримує два способи створення RESTful сервісів:

– із використанням MVCз об’єктом ModelAndView;

– за допомогою конвертерів HTTP-повідомлень.

У Hybris використовується саме другий, новіший підхід, заснований на HttpMessageConverter та анотаціях. Цей метод значно легший у підтримці та реалізації. Із мінімальною конфігурацією, розробник отримує інтуїтивно

зрозумілий каркас із усім необхідним для реалізації RESTful сервісів [19]. Перелік основних конвертерів у SAP Hybris Commerce:

– MappingJackson2HttpMessageConverter;

– StringHttpMessageConverter;

– JaxbOAuth2AccessTokenMessageConverter;

– ByteArrayHttpMessageConverter;

– JaxbOAuth2AccessTokenMessageConverter;

– JaxbOAuth2ExceptionMessageConverter.

Для розробки програмної системи Standby також використаний об’єктно-реляційний маппінґ (ORM) Hibernate. Поняття об’єктно-реляційного маппінґу означає автоматичнее і прозоре збереження (persistence-механізм)об’єктівуJava застосункахдотаблицьуSQLбазахданихізвикористанняммметадати, щоописує маппінґ між классами програми та схемою бази даних. Тобто задача ORM полягає у зворотньому перетворенні даних із однієї форми в іншу.

Причиною використання Hibernateє те, що плаформа Hybris передбачає підтримку великої кількості SQL-провайдерів. ORM відповідає на питання: як досягнути портативності, якщо кожна СУБД має власний SQL-діалект? До переваг Hibernate належать інноваційність, постійна оновлюваність, підтримка великої кількості СУБД, ефективність, open-source характер [20].

У розширенні Standbywebservices також використовується технологія JAXB (Java Architecture for XML Binding). Вона забезпечує швидкий і зручний спосіб запису Java-об’єктів у форматі XML або JSON і зворотню конвертацію.

JAXB підтримує фреймворк, який здійснює маппінґ XML-елементів і атрибутів у Java поля, використовуючи анотації [21]. У Hybris JAXB використовується для легкої автоматичної конвертації внутрішніх об’єктів DTO, які формуються на рівні фасадів, у зовнішні об’єкти WsDTO (Webservices Data Transfer Object), які використовуютьсяу HTTP Response.

Середовищем розробки обрана InteliJ IDEA Ultimate2019.x - містка та ергономічна IDE для JVM. Для студентів ХНУРЕ ліцензія Ultimate-версії надається безкоштовно. IDEA створена для підвищення продуктивності

програмування [22]. Вона забезпечує зручні засоби дебаггінгу та аналітики коду, інтеграцію із системами контролю версій (Git, Mercurial), СУБД, надає чудові шаблони рішень для веб-розробки, підримує користувацькі плагіни.

Для створення програмного рішення також використаний Hybris Integration плагін версії 8.x для InteliJ IDEA. Він забезпечує зручний імпорт Hybris-розширень із автоматичним вирішенням залежностей, оптимізовану компіляцію для юніт- та інтеграційних тестів, автоматичну комфігурацію для Spring, Ant, конекторів баз даних [23].

Плагін значно полегшує роботу з ImpEx-файлами та FlexibleSearch, валідацію дляфайлів маппінґу шару моделей items.xml та багато іншого.

4.3 Програмна реалізація

Для створення моделі даних у Hybris використовується XML-маппінґ, який конвертується засобами HybrisPersistance та ORM Hibernate у таблиці даних відповідно до конкретної обраної СУБД. Модель аудиту результатів показана на рисунку 4.4.

```
<itemtype code="StandbyAudit" extends="GenericItem" autocreate="true" generate="true">
    <deployment table="StandbyAudit" typecode="20052"/>
    <attributes>
        <attribute qualifier="importTime" type="java.lang.Long">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="exportTime" type="java.lang.Long">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="importCPULoad" type="java.lang.Double">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="exportCPULoad" type="java.lang.Double">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="methodName" type="java.lang.String">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="alternativeType" type="StandbyAlternativeType">
            <persistence type="property"/>
        </attribute>
        <attribute qualifier="checksumBefore" type="localized:java.lang.String">
            <persistence type="property"...>
        </attribute>
        <attribute qualifier="checksumAfter" type="localized:java.lang.String">
            <persistence type="property"...>
        </attribute>
    </attributes>
</itemtype>
```

Рисунок 4.4 – Модель сутності SearchAudit

Звернімо увагу, що для зберігання checksumBefore, checksumAfter, які містять закодовану у Sha1 строку цілісності об’єкту замовлення OrderModel у деяких базах даних може виявитися недостатньо стандартної довжини строки VARCHAR, тому Hybris дозволяє застосовувати різні типи колонок в залежності від СУБД:

```
<attribute qualifier="checksumBefore" type="localized:java.lang.String">
<persistence type="property">
<columntype database="oracle">
<value>CLOB</value>
</columntype>
<columntype database="mysql">
<value>TEXT</value>
</columntype>
<columntype database="sqlserver">
<value>TEXT</value>
</columntype>
<columntype database="sap">
<value>NCLOB</value>
</columntype>
<columntype>
<value>HYBRIS.LONG_STRING</value>
</columntype>
</persistence>
</attribute>
```

Таким чином досягається гнучкість і адаптивність шару моделей. Задачі генерації замовлень та імітації аварії створюються за допомогою ImpEx[24], вміст скрипту якого показаний на рисунку 4.5.

```
INSERT_UPDATE ServicelayerJob; code[unique]; springId
; standbyOrderFulfilmentJob  ; standbyOrderFulfilmentJobPerformable
; standbyDisaterImitationJob ; standbyOrderDisasterImitationPerformable

INSERT_UPDATE StandbyOrderFulfilmentCronJob[disable.interceptor.types = validate]; code[unique = true]; job(code); s
; ORDER GENERATION 1 HRS RPO  ; standbyOrderFulfilmentJob ; false ; en ; 1  ; 100
; ORDER GENERATION 2 HRS RPO  ; standbyOrderFulfilmentJob ; false ; en ; 2  ; 100
; ORDER GENERATION 4 HRS RPO  ; standbyOrderFulfilmentJob ; false ; en ; 4  ; 100
; ORDER GENERATION 8 HRS RPO  ; standbyOrderFulfilmentJob ; false ; en ; 8  ; 100
; ORDER GENERATION 12 HRS RPO ; standbyOrderFulfilmentJob ; false ; en ; 12 ; 100
; ORDER GENERATION 24 HRS RPO ; standbyOrderFulfilmentJob ; false ; en ; 24 ; 100

INSERT_UPDATE StandbyDisasterImitationCronJob[disable.interceptor.types = validate]; code[unique = true]; job(code);
; DISASTER IMITATION FULL       ; standbyOrderFulfilmentJob ; false ; en ; true
; DISASTER IMITATION PARTICULAR ; standbyOrderFulfilmentJob ; false ; en ; false
```

Рисунок 4.5 – ImpExдля створення задач Standby

Дані можуть бути використані на будь-якій із доступних СУБД.Для визначення чексуми та заміру рівня завантаження процесора реалізовані методи в абстрактному класі StandbyMeasurementAspect, який є батьківським классом для:

– ImpExManagerTimeMeasurementAspect;

– ImportExportServiceTimeMeasurementAspect;

– CommerceCBMTimeMeasurementAspect.

Розглянемо реалізацію методу для визначення Sha1-чексуми для моделі замовлення із класу StandbyMeasurementAspect:

```
protected String getOrdersChecksum() {
        final CatalogVersionModel catalogVersion =
catalogVersionService.getCatalogVersion(STANDBY_CATALOG_ID, ONLINE);

        return standbyOrderService.getOrders(catalogVersion).stream()
                .map(standbyOrderService::encode)
                .collect(Collectors.joining("\n"));
}
```

Логіка вимірювання відсотка завантаженості процесора:

```
protected double getProcessCpuLoad() throws InstanceNotFoundException,
ReflectionException, MalformedObjectNameException {
        MBeanServer mbs = ManagementFactory.getPlatformMBeanServer();
        ObjectName name =
ObjectName.getInstance("java.lang:type=OperatingSystem");
        AttributeList list = mbs.getAttributes(name, new
String[]{"ProcessCpuLoad"});

        if (list.isEmpty()) {
            return Double.NaN;
        }

        Attribute att = (Attribute) list.iterator().next();
        Double value = (Double) att.getValue();

        // returns a percentage value with 1 decimal point precision
        return value != -1.0
                ? ((int) (value * 1000) / 10.0)
                : Double.NaN;
    }
```

Результат виконання методів імпорту та експорту доступний як у графічному режимі Backoffice, так і звикористанням HTTP-запитів. Приклад реалізації REST-контролера для отримання статусу замовлень, генерованих системою, показаний на рисунку 4.6.

```
@Secured("ROLE_TRUSTED_CLIENT")
@RequestMapping(value = "/standby/orders", method = RequestMethod.GET)
@ResponseBody
@ApiOperation(value = "Get encoded ordews checksums.", notes = "Used to scrap all orders from Standby Test Catalog", authorizations =
        {@Authorization(value = "oauth2_client_credentials")})
@ApiBaseSiteIdAndCurrAndLangParam
public List<String> getStandbyOrders(@ApiParam(value = "Response configuration (list of fields, which should be returned in response)"

    final CatalogVersionModel catalogVersion = catalogVersionService.getCatalogVersion(STANDBY_CATALOG_ID, ONLINE_VERSION);
    final List<OrderModel> orders = standbyOrderService.getOrders(catalogVersion);
    return CollectionUtils.emptyIfNull(orders).stream()
            .map(standbyOrderService::encode)
            .collect(Collectors.toList());
}
```

Рисунок 4.6 – Реалізація REST-контролера

Також реалізований ендпоїнт REST-контролера для отримання результатів аудиту StandbyAuditModel відповідно до типу використаної альтернативи, часу виконання, ліміту кількості записів.

4.4 Тестування програмної системи

Після установки програмного модуля Standby як користувацького розширення проведене функціональне тестування, яке включає:

– перевірку цілісності первинних даних (initialdata), необхідних для роботи модуля;

– перевірку коректності роботи задачі генерації даних на невеликих обсягах даних;

– перевірку коректності роботи задачі порушення цілісності даних;

– верифікацію аспектів та коректності створених моделей аудиту;

– перевірку ендпоїнтів REST-контролера для отримання даних про замовлення та списку результатів аудиту для кожної із використаної альтернативи.

Перевірка цілісності даних проводиться шляхом мануального тестування при виконанні AdvancedSearch у Backoffice Administration перспективі, як

показано на рисунку 4.6. Очікуваний результат:каталог Standby Test Catalog існує, наявний тестовий набір із продуктів у каталог версії StandbyTestCatalog: Online.

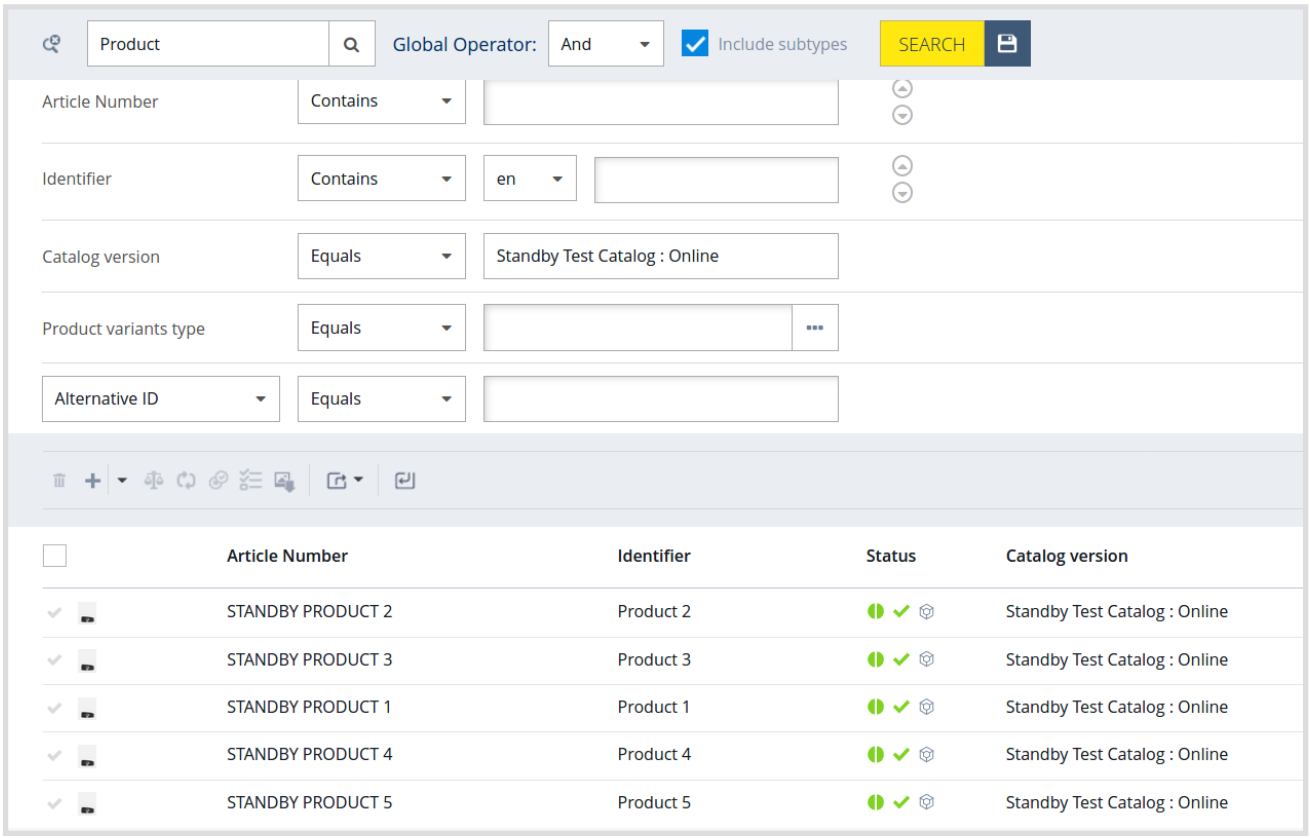

Рисунок 4.6 – Тестування цілісності даних за допомогою AdvancedSearch

Продукти повинні бути видимі у веб (visibleinWeb), мати сконфігуровані ціни у євро (PriceRowModel), а також принаймні 1 медіа-об’єкт (MediaModel). Результати мануального тестування показані на рисунках 4.7 та 4.8.

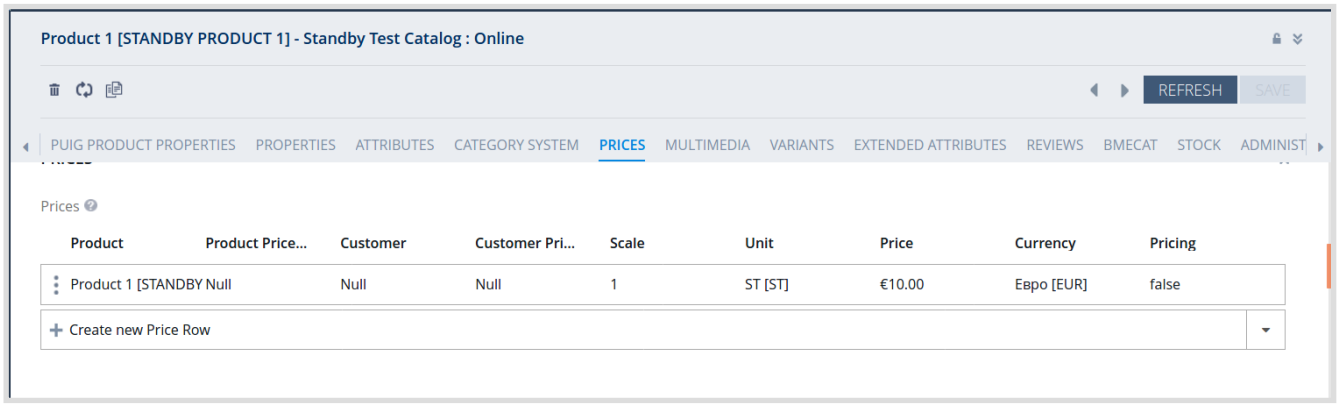

Рисунок 4.7 – Тестування цілісності об’єктів PriceRow

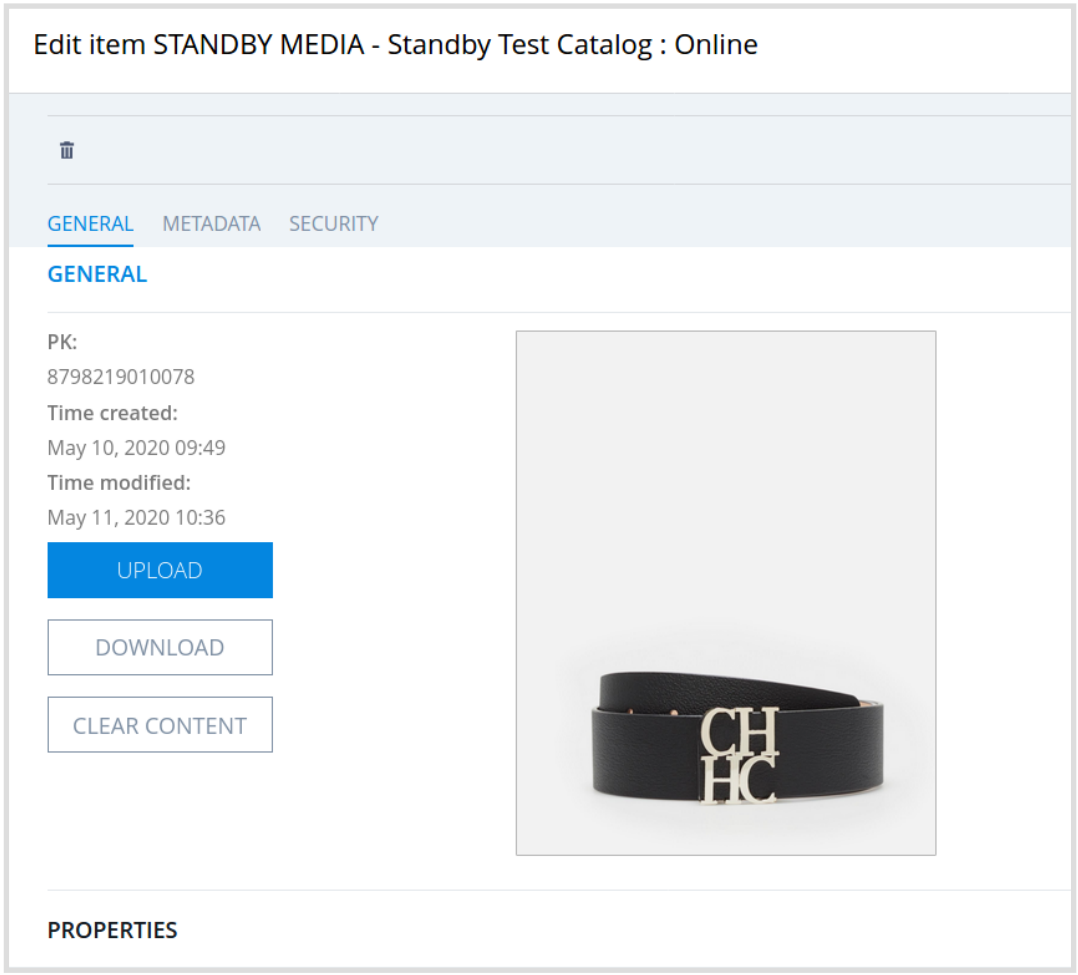

Рисунок 4.8 – Тестування цілісності об’єкту MediaModel

Перевірка функціонування задачі StandbyOrderFulfilmentCronJob виконується шляхом її запуску з конфігурацією змінних: RTO =1 година, hourllyAmount = 10. Очікуваний результат: 10 нових замовлень OrderEntry згенеровані у системі. Кожне замовлення має користувача та рандомно сформований список OrderEntryModel, кожна із яких містить один із наявних у каталозі StandbyTestCatalog продуктів, та кількість quantity. Для перевірки кількості виконується FlexibleSearch-запиту середовищі HAC:

```
"SELECT COUNT({o."+ AbstractOrderModel.PK + "}) FROM {"
+ OrderModel._TYPECODE + " AS o "
+"JOIN " + AbstractOrderEntryModel._TYPECODE + " AS oe ON {oe."
+AbstractOrderEntryModel.ORDER + "} = {o." + AbstractOrderModel.PK + "} "
+"JOIN " + ProductModel._TYPECODE + " AS p ON {oe."
+AbstractOrderEntryModel.PRODUCT + "} = {p." + ProductModel.PK + "} }"
+ "WHERE {p." + ProductModel.CATALOGVERSION + "} = ?catalogVersion "
+"GROUP BY {o." + AbstractOrderModel.PK + "}"
```

Тестування функціоналу задачі StandbyDisasterImitationCronJob відбувається за допомогою двох типів задач, спершу із значенням fullCorruption, встановленим у false (часткове пошкодження), після цього - навпаки. Очікуваний реультат: для часткового пошкодження FlexibleSearch-запит поверне значення 0, оскільки зв'язки між замовленням та одиницями замовлення і продуктами зруйновані. При цьому самі об'єкти залишаються доступними для AdvancedSearch. Для другого сценарію у системі очікується повне стирання тестових об'єктів OrderModel.

Успішне проходження функціонального тестування і виконаний на 100% набір юніт-тестів із директорії standbycore/testsrc, який виконується у junit-тенанті, означає готовність програмної системи до використання з метою дослідження методів створення бекапів (експорту) та післяаварійного відновлення (імпорту) з використанням цих ресурсів.

ВИСНОВКИ

У роботі були проаналізовані методи, моделі та програмні засоби резервного копіювання та післяаварійного відновлення даних, а сааме після аварійного відновлення як послуги DRaaS у рамках потреб електронної комерції для популярної платформи SAP Hybris Commerce. Дослідження стану проблеми показало, що стрімкий зріст обсягів сховищ даних пов’язаний із потребами бізнесу eCommerce. Водночас IT-компанії визнають незадовільним стан інфраструктури для здійснення резервних копій важливої інформації.

Моніторинг стану проблеми та наявних методів і моделей резервного копіювання та післяаварійного відновлення в Hybris підтвердив необхідність розробки гнучкої системи оцінювання ефективності даних методів.

У роботі запропонована методика та визначені критерії оцінювання, найважливішими серед яких є об’єктивний час відновлення RTO, ступінь завантаженості процесора, кількість звернень до бази даних. Виявлені фактори впливу, серед яких частота створення резервних копій тапотужність сервера.

Спроектований і реалізований модуль Standby для SAP Hybris Commerce, який дозволив виділити найбільш ефективні методи створення резервних копій із подальшим їх відновленням. Серед них - технологія ImpEx, яка надається розробниками платформи OOTB, а також розширення із відкритим сирцевим кодом Aimprosoft CBM for SAP Commerce.

Проведена у рамках роботи низка експериментів дозволила виявити такі основні концепти: технологія ImpEx показує найкращі показники щодо RTO та продуктивності сервера при будь-яких RPO. Водночас отримані результати для розширення Aimprosoft Cloud Backup Migration наближені до OOTB-показників, що свідчить про високу ефективність програмного продукту. Відмінності у часі роботи можна пояснити наявністю додаткових генераторів скриптів, які значно полегшують роботу менеджера з бекапу, та шаром моделей, які зберігають

інформацію про виконані задачі імпорту та експорту й підвищують надійність системи.

При високих (4-8 годин і більше) значеннях RPO показники СВМ практично зрівнюються із аналогічними замірами для ООТВ-технології. Враховуючи функціональну складову, цей метод є рекомендованим при виборі стратегії післяаварійного відновлення. І навпаки, при використанні частих RPO (менш ніж 4 години), рекомендовано генерувати скрипти вручну й використовувати традиційний підхід до імпорту та експорту, без операцій зберігання метаданих про результати роботи методів.

У ході роботи підтверджена гіпотеза про наявність точки балансу між витратами через втрату даних та витратами на створення бекапів. Такою точкою у контексті даного тестового середовища стала RPO, що дорівнює восьми годинам, яка є оптимальною для збереження відмовостійкості системи без значної шкоди її швидкодії. Рекомендується також використання транзакцій, оскільки вони майже не знижують потужності роботи аналізованих сервісів, однак гарантують безпечність та ізольованість виконання методів.

Було доведено, що система Hybris має потужну базу для розробки, є швидкою і надійною, використовує найкращі практики і технології корпоративного програмування. У всіх проведених експериментах час RTO є незначним у порівнянні із RPO. Це означає, що платформа здатна до швидкого відновлення свого стану навіть в умовах пікових навантажень.

Розроблена програмна система Standby є продуктом із відкритим сирцевим кодом і розміщена на ресурсі GitHub. Результати оцінювання ефективності і рекомендації щодо створення плану післяаварійного відновлення серверних даних опубліковані в статті «Дослідження ефективності методів і моделей резервного копіювання та післяаварійного відновлення даних у SAPHybrisCommerce» в міжнародному науковому журналі «Science Online».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Gai S. Building a Future-Proof Cloud Infrastructure: The Unified Architecture for Network, Security, and Storage Services. - Boston, MA: Addison-Wesley, 2020, 118 p.

2. Harvard Business Review Press. HBR Guide to Data Analytics Basics for Managers. - Boston, MA: Harvard Business Review Press, 2018, 256 p.

3. SAP Hybris Development: tips, tricks and developers cost // Blog / Aimprosoft. URL: https://www.aimprosoft.com/blog/sap-hybris-development-tips-tricks-and-developers-cost(дата звернення: 12.03.2020).

4. SAP Hybris - сімейство продуктів // ПідручникиSAP/ CoderLessons. URL: https://coderlessons.com/tutorials/sap/izuchite-sap-hybris/sap-hybris-tutorial (дата звернення: 16.03.2020).

5. Stevens B. DRaaS basics: Veeam special edition. - Baar: Veeam Press, 2015, 110 p.

6. Future-Proofing Your Data Center Storage // Technology Adoption Profile / IT World Canada. URL: https://s3-us-west-2.amazonaws.com/itworldcanada/archive/Documents/whitepaper/ITW240B_Future_Proofing.pdf (датазвернення: 21.04.2020).

7. Пупена О. Хмарні сервіси в АСУТП // Технології індустрії 4.0 / Школа автоматики. URL: http://edu.asu.in.ua/mod/book/view.php?id=119 (дата звернення: 12.04.2020).

8. Disaster Recovery as a Service (DRaaS) // Безпека / Data Park. URL: https://datapark.com.ua/ua/services/zahyst/draas (дата звернення: 12.04.2020).

9. Колдовський В. Веб-розробка: вчора, сьогодні, завтра// Стрічка новин/ DOU. URL: https://dou.ua/lenta/articles/web-development-status-2020 (дата звернення: 12.03.2020).

10. Hybris Usage Statistics // eCommerce / Trends BuiltWith. URL: https:// trends.builtwith.com/shop/Hybris/Market-Share (дата звернення: 03.04.2020).

11. SAP Hybris Cluster: Usage and Limitations // Performance Zone / DZone. URL: https://dzone.com/articles/sap-hybris-cluster-usage-and-limitations (дата звернення: 03.04.2020).

12. Architecture of SAP Commerce: Jalo Layer // Platform / SAP Help Portal. URL: https://help.sap.com/viewer/d0224eca81e249cb821f2cdf45a82ace/1808/en-US/8c00066686691014a5a5d19875a1525b.html (дата звернення: 12.04.2020).

13. Jalo Layer // Hybris Commerce Development / A Techie Talk. URL: https://mjyothula.wordpress.com/2017/01/17/jalo-layer (дата звернення: 12.04.2020).

14. Machnev A. Functional Specification Document: Hybris Commerce CBM v.1.0. - Kharkiv: Aimprosoft, 2020, 4p.

15. Плани та тарифи хостингу виділеного сервера // Сервери / GoDaddyУкраїна. URL: https://ua.godaddy.com/hosting/dedicated-server (дата звернення: 10.04.2020).

16. Мазурова О. Прийняття рішень в умовах визначеності // Теорія планування та прийняття рішень / ХНУРЕ ДН. URL: https://dl.nure.ua/pluginfile.php/142440/mod_resource/content/2/ТПиПР_лк_3_укр.pdf (дата звернення: 22.04.2020).

17. Schildt H. Java: The Complete Reference: Ninth Edition. - New York City, NY: McGraw-Hill Osborne Media, 2014, 1312p.

18. Hemrajani A. Agile Java Development with Spring, Hibernate and Eclipse. - Indianapolis, IN: Sams Publishing, 2006, 128p.

19. Build a REST API with Spring and Java Config // Guides / Baeldung. URL: https://www.baeldung.com/building-a-restful-web-service-with-spring-and-java-based-configuration (двта звернення: 30.03.2020).

20. Bauer C., King G., Gregory G. Java Persistence with Hibernate: Second Edition. - Shelter Island, NY: Manning Publications Co., 2016, 610p.

21. Guide to JAXB // Guides / Baeldung. URL: https://www.baeldung.com/jaxb(дата звернення: 04.04.2020).

22. Capable and Ergonomic IDE for JVM // InteliJ IDEA / JetBrains. URL: https://www.jetbrains.com/idea/promo/ultimate(дата звернення: 11.04.2020).

23. Integration for SAP Commerce // Framework Integration / JetBrains. URL: https://plugins.jetbrains.com/plugin/12639-integration-for-sap-commerce(дата звернення: 11.04.2020).

24 Ніконова А., Лєсна Н.. Дослідження ефективності методів і моделей резервного копіювання та післяаварійного відновлення даних у SAP Hybris Commerce. International Electronic Scientific Journal «Science Online», 2020