https://orcid.org/0000-0002-2552-2839
https://orcid.org/0000-0001-5568-7629
https://orcid.org/0000-0002-4133-722X

УДК 004.02

С.В. ГРИБКОВ[*], В.А. ЛИТВИНОВ[**], Г.В. ОЛІЙНИК[*]

ІНСТРУМЕНТАЛЬНА МОДЕЛЬ ВЕБ-ОРІЄНТОВАНОЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ ПІДСИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ У СКЛАДІ ПРОГРАМНОГО КОМПЛЕКСУ СИТУАЦІЙНОГО ЦЕНТРУ

[*]Національний університет харчових технологій, м. Київ, Україна
[**]Інститут проблем математичних машин і систем НАН України, м. Київ, Україна

Анотація. *Одним із важливих завдань при вирішенні проблеми створення мережі Ситуаційних центрів (СЦ) органів державної влади України є відпрацювання типових рішень у різних сферах проблематики проектування, впровадження і функціонування СЦ. Метою статті є спроба часткового заповнення деяких наявних прогалин у відношенні методологічного та інструментального забезпечення реалізації програмного комплексу СЦ. Відзначається беззаперечна доцільність застосування ітераційного (еволюційного) підходу до процесу проектування СЦ з урахуванням реальних вітчизняних умов та ризиків. Як можлива основа прототипування СЦ розглядаються сучасні ECM-системи, що визначаються як програмні комплекси управління корпоративним контентом. У рамках цих підходів пропонується інструментальна модель прототипу підсистеми підтримки прийняття рішень у складі програмного комплексу СЦ, зокрема, на етапах підготовки і проведення наради. Під інструментальною моделлю розуміється сукупність програмних компонентів підсистеми і рішень щодо їх спільного застосування. Модель веб-орієнтованої програмної реалізації підсистеми включає програмну платформу Spring Framework, що загалом забезпечує побудову програмної інфраструктури підсистеми; засоби бібліотеки Junit та програмної платформи Mockito для модульного та інтеграційного тестування програмного коду; засоби об'єктно-реляційного відображення Hibernate для роботи з даними; протокол HTTP Request-Response як основу взаємодії серверної частини із клієнтською; стандарт/специфікацію Open API опису інтерфейсів взаємодії між серверною та клієнтською частинами; засоби формування web-сторінок для відображення у браузері користувача. Сукупність програмно-технологічних рішень, представлених у моделі, успішно апробовано в системі підтримки прийняття рішень щодо формування і оперативної реконфігурації виробничих планів виконання договорів підприємства.*

Ключові слова: *ситуаційний центр (СЦ), прототипування програмних комплексів СЦ, ECM-системи, інструментальна модель програмної реалізації.*

Аннотация. *Одной из важных задач в решении проблемы создания сети Ситуационных центров (СЦ) органов государственной власти Украины является отработка типовых решений в различных сферах проблематики проектирования, внедрения и функционирования СЦ. Целью статьи является попытка частичного заполнения некоторых имеющихся пробелов в отношении методологического и инструментального обеспечения реализации программного комплекса СЦ. Отмечается несомненная целесообразность применения итерационного (эволюционного) подхода к процессу проектирования СЦ с учетом реальных отечественных условий и рисков. В качестве возможной основы прототипирования СЦ рассматриваются современные ECM -системы, которые определяются как программные комплексы управления корпоративным контентом. В рамках этих подходов предлагается инструментальная модель прототипа подсистемы поддержки принятия решений в составе программного комплекса СЦ, в частности, на этапах подготовки и проведения совещания. Под инструментальной моделью понимается совокупность программных компонентов подсистемы и решений по их совместному применению. Модель веб-ориентированной программной реализации подсистемы включает программную платформу Spring Framework, которая в целом обеспечивает построение программной инфраструктуры подсистемы; средства библиотеки Junit и программной платформы Mockito для модульного и интеграционного тестирования программного кода; средства объектно-реляционного отображения Hibernate для работы с данными; протокол HTTP Request-Response как основу взаимодействия серверной части с клиентской; стандарт/спецификации Open API описания интерфейсов взаимодействия между серверной и клиентской частями; средства формирования web-страниц для отображения в браузере пользователя. Совокупность программно-технологических решений,*

ISSN 1028-9763. Математичні машини і системи, 2020, № 1

представленных в модели, успешно апробированы в системе поддержки принятия решений по формированию и оперативной реконфигурации производственных планов выполнения договоров предприятия.
Ключевые слова: *ситуационный центр (СЦ), прототипирование программных комплексов СЦ, ECM-системы, инструментальная модель программной реализации.*

Abstract. *One of the most important tasks in solving the problem of creating a network of situational center (SC) of public authorities in Ukraine is to develop standard solutions in various areas of the problems of design, implementation and functioning of the SC. The purpose of the article is an attempt to partially fill some of the gaps in terms of methodological and instrumental support for the implementation of the SC software. It is undoubtedly expedient to apply an iterative (evolutionary) approach to the process of designing a SC taking into account real domestic conditions and risks. As a possible basis for prototyping the SC are considered modern ECM-systems, which are defined as software complexes of corporate content management. These approaches propose an instrumental model of the prototype of a decision support subsystem as part of the SC complex, in particular during the preparatory and meeting stages. The instrumental model is understood as a set of software components of the subsystem and decisions on their joint application. The web-based software implementation model of the subsystem includes: Spring Framework software platform, which in general provides the construction of software infrastructure of the subsystem; means of Junit library and Mockito software platform for modular and integration testing of the code of the developed system; Hibernate object-relational mapping tools for working with data; HTTP Request-Response protocol as the basis for interaction between the server and the client side; standard / Open API specifications describing the interfaces between the server and client parts; means of forming web-pages for displaying them in the user's browser. The set of software and hardware solutions presented in the model has been successfully tested in the decision support system for the formation and operational reconfiguration of production plans for the implementation of contracts of the enterprise.*
Keywords: *situational center (SC), prototyping of SC software systems, ECM-system, software instrumental model.*

DOI: 10.34121/1028-9763-2020-1-73-81

1. Вступ

В останні роки Ситуаційні центри (СЦ) набувають все більш широкого розповсюдження як інтелектуальний інструмент підтримки прийняття ефективних рішень для складних слабодетермінованих аналітичних задач. Особливо велику роль вони відіграють як засіб інтелектуального забезпечення державного управління життєдіяльністю суспільства взагалі і національною безпекою зокрема [1-3]. Вирішення проблем координації діяльності та взаємодії органів державної влади України на загальнодержавному та міжнародному рівнях має забезпечити створення мережі СЦ [4]. Одним із найбільш важливих завдань у зв'язку з цим стає підготовка типових рішень у різних сферах проблематики створення, впровадження і функціонування СЦ.

Наявні вітчизняні і доступні іноземні публікації за проблематикою СЦ присвячені переважно загальним концептуальним питанням: структурним, функціональним, технологічним аспектам побудови та використання СЦ, принциповим рішенням щодо організації інформаційного забезпечення СЦ, створення єдиного інформаційного простору мережі СЦ та ін. Питання належної програмної реалізації підсистем та окремих функцій СЦ залишаються, поки що, якось «на узбіччі».

Метою даної статті є спроба як внесення посильного вкладу в часткове заповнення відзначеної прогалини, зокрема, у відношенні методологічного та інструментального забезпечення реалізації програмного комплексу СЦ, так і привернення до цієї проблематики додаткової уваги фахівців-проектантів і відповідних стейкхолдерів.

2. ECM-системи як можлива основа прототипування СЦ

Сучасні методології і технології створення складних програмних систем (RUP, MSF, XP тощо) засновані на ітераційному (еволюційному) підході до побудови моделі процесу проектування. Стосовно до умов і реальних вітчизняних можливостей особливої уваги заслуговує спіральна різновидність еволюційної моделі, орієнтована на урахування відповідних десяти ризиків, сформульованих Б. Боемом (дефіцит спеціалістів та розрив між їхньою кваліфікацією й вимогами проекту, нереалістичні строки та бюджет тощо) і заснована на прототипуванні фрагментів (підсистем) або версій програмного комплексу. В контексті проблеми прототипування та розробки типових рішень по створенню програмних комплексів СЦ відзначимо, що на теперішній час у сфері інформаційних технологій склалася концепція ECM-систем (Enterprise Content Management), що визначаються і позиціонуються як програмний комплекс управління корпоративним контентом, призначений, з одного боку, для створення єдиного інформаційного простору різноманітних документів підприємства, а з іншого - для управління процесами їх обробки та надання користувачам. У теперішній час серед IT-спеціалістів та експертів з управління корпоративними системами поки що так і не склалося єдиної думки щодо термінології і конкретних визначень функцій ECM-систем. Більш того, вважається, що ECM – це деякий «коктейль, рецептура якого може відрізнятися від бара до бару» [5] . Проте всі визначення, в тому числі і визначення відомої дослідницької та консалтингової фірми Gartner, що спеціалізується на ринку IT, в основні функції включають управління документами, мультимедіаконтентом (графічними, аудіо- та відеофайлами), знаннями (knowledge management), потоками робіт (workflow management), потоками документів (docflow management), спільною роботою користувачів із документами (collaboration management).

Окремим обмеженим випадком ECM-систем є системи електронного обігу документів (СЕД), позбавлених цілої низки важливих функцій, зокрема, workflow management, knowledge management, collaboration та ін. Відзначимо, що виділення даної категорії систем (як і поняття «документообігу») прийнято у пострадянському просторі, де поняття ECM-систем найчастіше збігається з поняттям СЕД [6, 7].

Розглядаючи програмний комплекс Ситуаційного центру як розширену ECM-систему специфічного «підприємства» (а для цього, на наш погляд, є достатньо підстав), відзначимо, по-перше, що саме відзначені вище функції є одними з найбільш важливих для цілей створення СЦ, а по-друге, що архітектурні підходи і програмні рішення, прийняті в наявних ECM-системах, можуть, як мінімум, бути корисними для створення програмного комплексу повнофункціональних СЦ.

У теперішній час ринок класичних ECM-систем для «звичайних» корпорацій нараховує ряд конкретних систем, що пропонуються різними виробниками, в тому числі і російськими. Десять найбільш популярних систем описані і охарактеризовані в [8]. При побудові наведених ECM-систем використовуються різні архітектурні рішення, порівняно простим прикладом яких, близьким до структурних рішень СЦ, може бути загальна архітектура ECM фірми «Галактика» [9] (рис. 1). Основою для розширеної конкретизації складу функціональних підсистем програмного комплексу ECM СЦ порівняно з наведеним прикладом може слугувати орієнтовна типова структура процесів СЦ, рекомендована в [10].

У контексті задачі прототипування і відпрацювання типових проектних рішень СЦ як специфічної ECM-системи можна відзначити два можливих підходи:

– адаптація і використання готових free и open-sourse ECM-платформ [9, 11];

– розробка на базі ідеології і апробованих архітектурних рішень ECM спеціалізованих програмних систем і моделюючих комплексів, орієнтованих безпосередньо на задачі і особливості СЦ.

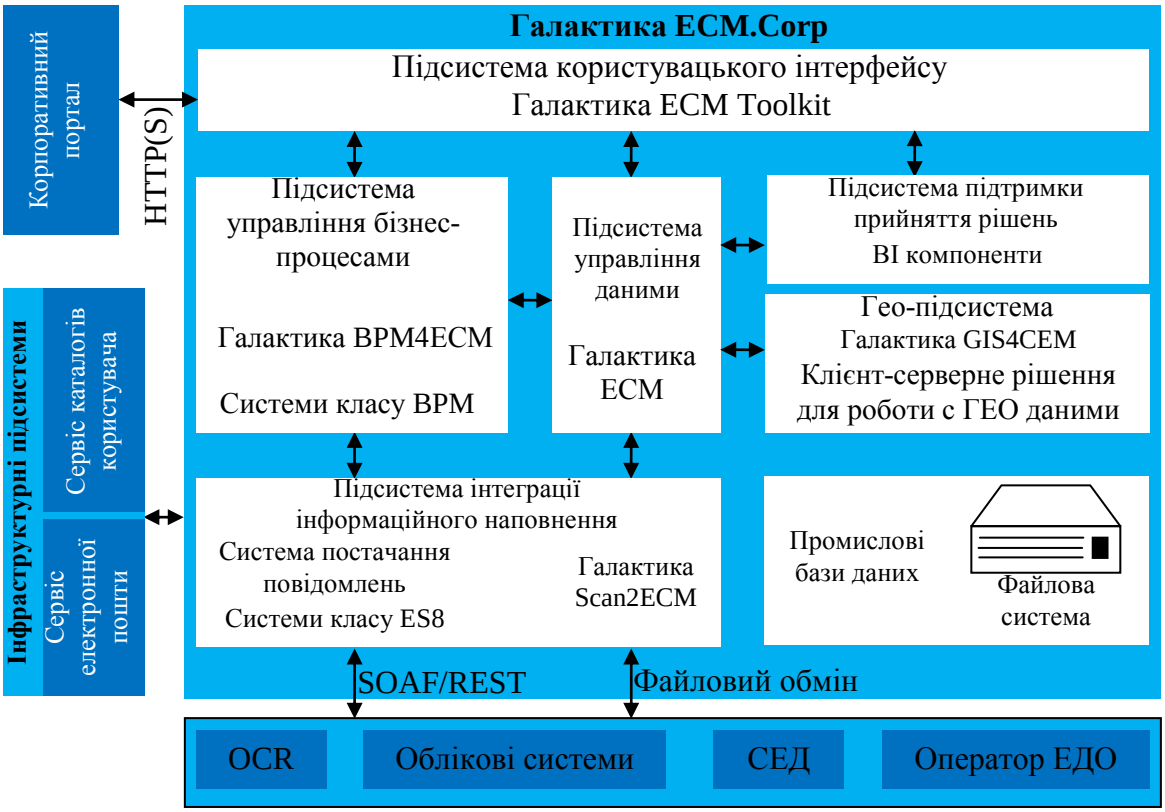

Рисунок 1 – Функціональна структура ECM «Галактика ECM.Corp»

Перший підхід потребує окремого розгляду і спеціальних, досить масштабних і ресурсомістких досліджень. У рамках другого підходу, як деякий крок в step_by_step-процесі створення типових програмних комплексів, ми розглянемо можливу інструментальну модель прототипу веб-орієнтованої програмної реалізації підсистеми підтримки прийняття рішень у складі сервера СЦ, зокрема, на етапах підготовки і проведення наради [10]. Відзначена підсистема є основою виконання функції колективного обговорення проблеми (collaboration), а під інструментальною моделлю тут розуміється сукупність програмних компонентів підсистеми і рішень щодо їх спільного застосування. Використання саме веб-інтерфейсу обґрунтовується рядом факторів, серед яких не останнє місце посідає необхідність роботи у процесі наради з віддаленими учасниками-експертами.

3. Модель програмної реалізації підсистеми

До основних компонентів системи, що розглядається, належать інформаційні ресурси СЦ, серверна частина, web-інтерфейс користувача (рис. 2).

Рішення щодо побудови програмної серверної частини базуються на класичній багатошаровій архітектурі з розподіленням на такі складові:

– шар контролерів, за допомогою яких забезпечується взаємодія із клієнтською частиною;

– адаптери, на рівні яких здійснюється перетворення даних із запитів клієнтів у основні бізнес-об’єкти, над якими здійснюється подальша обробка;

– класи, що здійснюють перевірки вхідних даних на відповідність встановленим правилам;

– бізнес-сервіси, які безпосередньо відповідають за виконання бізнес-логіки;

– допоміжні інструментальні класи, яким бізнес-сервіси делегують виконання окремих операцій;

– репозиторії, призначені для безпосередньої роботи з даними.

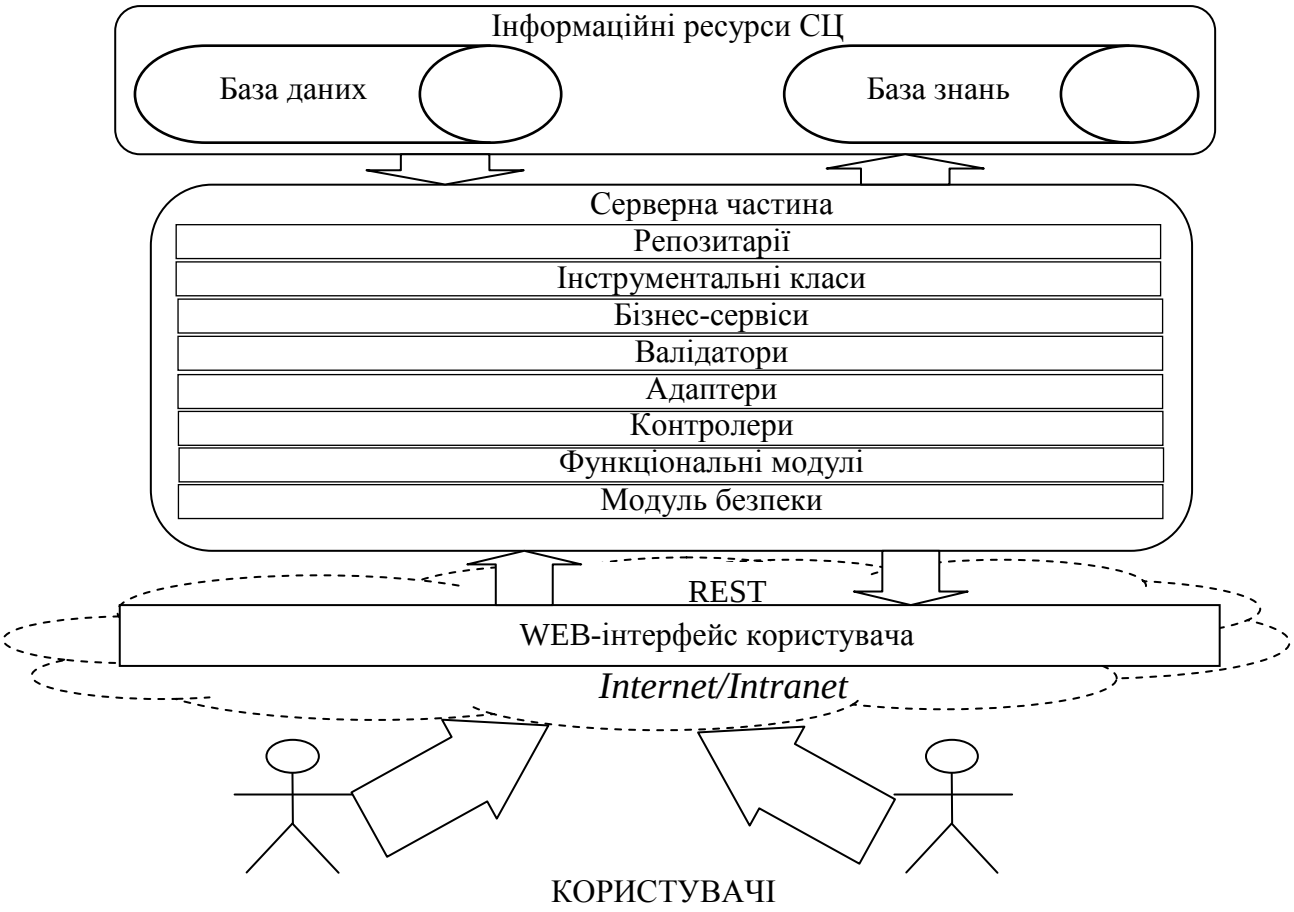

Рисунок 2 – Структура системи

Функціональні можливості серверної частини відповідають потребам користувачів і у загальному вигляді розподілені на модулі, призначені для вирішення конкретних задач, що реалізують функціонал системи, зокрема, на етапах підготовки і проведення наради. Кожен модуль при виконанні обробки деякого запиту звертається для отримання даних до інформаційних ресурсів системи. Після проведення обробки даних та необхідних підрахунків результат відображається клієнтською частиною у заданому вигляді. Кількість модулів не регламентується, чим забезпечується, у випадку необхідності, модернізація існуючих та включення нових модулів для вирішення задач прийняття рішень.

Розробка програмної логіки роботи модулів на серверній частині здійснюється мовою програмування Java, вибір якої зумовлений відомими перевагами [12], з використанням Spring Framework та Hibernate. Незважаючи на те, що мова програмування Java надає значну кількість вбудованих функціональних можливостей для розробки інформаційних систем, засоби для організації складових програмних частин складної системи у єдине ціле не надаються. Вирішення такої проблеми покладається безпосередньо на розробників та архітекторів. Для розв'язання даної задачі обрана програмна платформа Spring Framework, що надає формалізовані засоби об'єднання розрізнених компонентів у єдину програмну систему на основі принципу введення залежностей (Dependency Injection) [13]. У відповідності із вказаним принципом задача побудови залежностей об'єкта вирішується за допомо-

гою зовнішнього механізму. Ключове призначення програмної платформи Spring Framework полягає у забезпеченні побудови програмної інфраструктури системи.

Spring Framework має модульну структуру, яка дозволяє використовувати лише необхідні для реалізації функцій системи елементи [14].

Модулі об’єднані у групи за функціональним призначенням (рис. 3), а саме:

– ядро програмної платформи (Core Container);

– модулі доступу до даних (Data Access/Integration);

– модулі для розробки Web-орієнтованих систем (Web);

– модулі для роботи з аспектно-орієнтованим програмуванням (AOP – Aspect Oriented Programming);

– інструментальні модулі (Instrumentation);

– тестові модулі (Test).

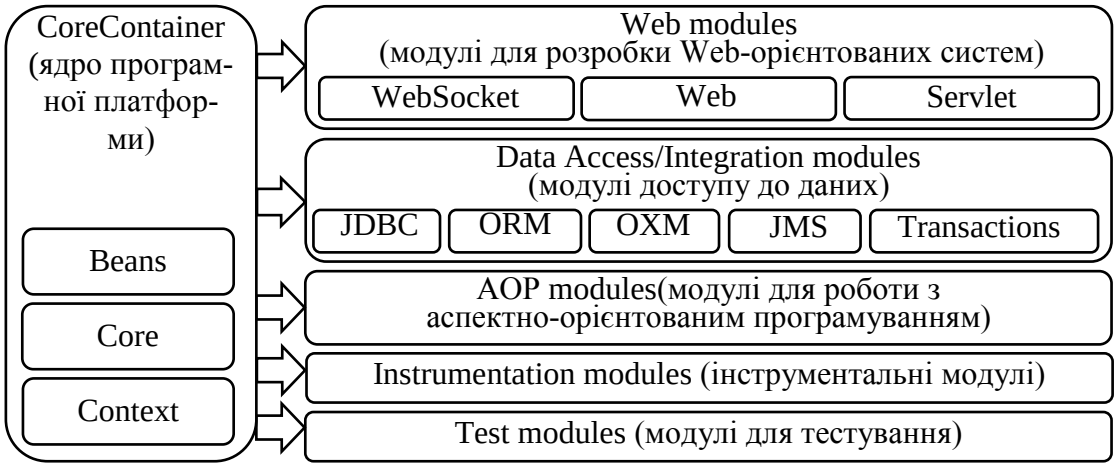

Рисунок 3 – Модулі програмної платформи Spring Framework, розподілені на групи за функціональним призначенням

Фундаментальну частину програмної платформи Spring Framework складають модулі Beans та Core, які забезпечують можливості введення залежностей між компонентами серверної частини системи СППР. Саме це дозволяє відокремити конфігурацію системи та специфікацію залежностей об’єктів. Модуль Context розширяє Beans, надає вбудовану підтримку такої функціональності:

– локалізація та інтероперабельність із використанням пакетів ресурсів;

– використання та поширення подій між компонентами для керування поведінкою;

– створення програмних середовищ (контекстів) для управління часом існування та взаємодією компонентів (наприклад, контейнер сервлетів).

Модуль JDBC надає власний рівень абстракції для безпосередньої роботи з інформаційними ресурсами, що забезпечує зменшення обсягів роботи для розробників, а також виключає необхідність обробки специфічних помилок від конкретних систем управління базами даних. Модуль ORM представляє собою основу для роботи з технологіями для об’єктно-реляційного відображення, такими як Hibernate. Модуль для обміну повідомленнями (JMS) надає функції для створення та отримання повідомлень, чим забезпечується взаємодія зі сторонніми інформаційними системами. Модуль Transactions підтримує програмне та декларативне керування транзакціями, що надає можливості для гнучкого керування цілісністю даних, використовуваних системою, та єдність операцій, які виконуються. Spring забезпечує роботу з локальними та глобальними транзакціями без прив’язки до технологій реалізації. Надається досить широка підтримка реалізації декларативних транзакцій як за допомогою XML-конфігурації, так і за допомогою анотацій. Крім цього, є мо-

жливість управляти транзакціями програмно, що зумовлює гнучкість при налаштуванні в залежності від конкретних вимог до функціональності системи.

Модулі для розробки Web-орієнтованих систем надають основні можливості розробки контролерів Web-сервісів для Web-інтеграції системи, що є ключовим для взаємодії серверної частини з користувальницькою.

Модуль AOP відповідає за роботу з аспектно-орієнтованим програмуванням та забезпечує можливість створення методів-перехоплювачів. Такі методи є досить корисними за необхідності виокремлення наскрізної для різних об'єктів функціональності. Також, можливості аспектно-орієнтованого програмування використовуються для додаткового ведення обліку інформації про усі операції, що здійснюються під час роботи системи, з описом усіх викликів та значеннями переданих аргументів. Крім цього, надається можливість інтеграції з бібліотекою AspectJ, яка надає розширені способи перехоплення викликів та впровадження деякої програмної логіки на різних етапах взаємодії з об'єктами.

Модульне та інтеграційне тестування програмного коду розроблюваної системи здійснюється з використанням:

– модулів для тестування компонентів під управлінням Spring Framework у поєднанні з бібліотекою для модульного тестування Junit;

– програмної платформи Mockito, яка дозволяє створювати об'єкти-заглушки для ізольованої перевірки найменших фрагментів коду у відповідності із принципами підходу модульного тестування.

Вибір програмної платформи для побудови архітектури інформаційної системи обумовлений можливістю повної незалежності бізнес-логіки від елементів програмної платформи та ізольованістю залежностей на компоненти від решти програмного коду. Дана перевага надається Spring Framework, що також вплинуло на її вибір як програмної платформи для побудови програмної архітектури.

Особлива увага має приділятися роботі із джерелами даних, оскільки взаємодія між об'єктами об'єктно-орієнтованої мови програмування та реляційною базою даних громіздка та трудомістка, що зумовлено, як відомо, невідповідністю парадигм зберігання даних, представлених класами об'єктів і таблицями баз даних. Для зв'язування баз даних з концепціями об'єктно-орієнтованих мов програмування доцільним є використання технології ORM (Object-Relational Mapping – об'єктно-реляційне відображення). Концепція ORM у Java EE представлена специфікацією JPA (Java Persistence API).

Для роботи з даними використовується одна з реалізацій підходу об'єктно-реляційного відображення, а саме Hibernate. Задачею Hibernate є перенесення даних із рядків таблиць бази даних у об'єкти Java і навпаки [15]. Крім цього, Hibernate надає вбудовані можливості для запитів і пошуку, що дозволяє скоротити час розробки, який витрачається на ручну обробку даних в SQL і JDBC. Проте, на відміну від багатьох подібних рішень, Hibernate не приховує потужності SQL і дозволяє за необхідності використовувати реляційні технології напряму. Вибір Hibernate також зумовлено відсутністю необхідності у складній взаємодії з базою даних, використанням складних запитів тощо, оскільки оперування даними відбувається на програмному рівні. У порівнянні з аналогами (наприклад, MyBatis), Hibernate відразу пропонує готове рішення, початкові вимоги до конфігурування якого мінімальні.

Загальну схему взаємодії рівня доступу до даних інформаційної системи з реляційною базою даних із використанням Hibernate зображено на рис. 4.

Взаємодія між кодом інформаційної системи та функціональними можливостями Hibernate відбувається через стандартний інтерфейс доступу JPA, а також через розширений інтерфейс Hibernate, який надає додаткові можливості.

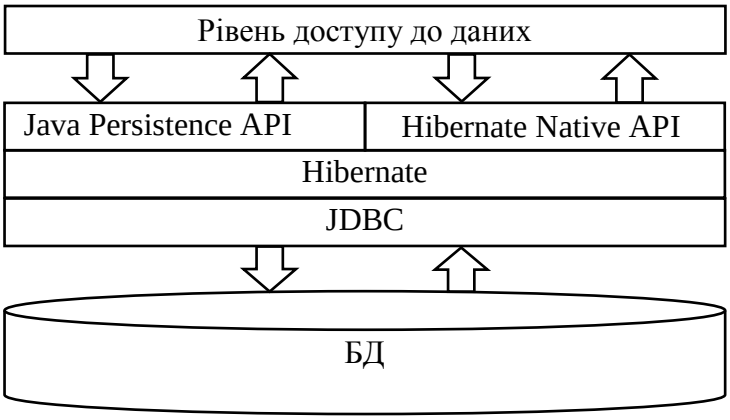

Рисунок 4 – Схема взаємодії рівня доступу до даних із використанням Hibernate

Взаємодія серверної частини із клієнтською здійснюється мережею Інтернет/Intranet за протоколом HTTP Request-Response з використанням ідеології архітектурного стилю для розподілених систем REST (Representational State Transfer), що дозволяє відокремити логіку системи від інтерфейсу користувача, а структуру системи зробити більш простою та масштабованою [16].

Для формалізованого опису інтерфейсів взаємодії між серверною та користувальницькою частиною використовується специфікація Open API, із застосуванням якої описуються усі ресурси, що надаються серверною частиною, та доступні операції над ними. Фактично, дана специфікація визначає стандарт опису REST інтерфейсів, який дозволяє взаємодіяти з серверною частиною без доступу до вихідного коду, додаткової технічної документації або прямої взаємодії через мережу [17].

Розробка програмного коду здійснюється з використанням підходу неперервної інтеграції CI (Continuous Integration) та програмного продукту Jenkins [18]. При їх використанні кожна зміна програмного коду автоматично фіксується та тестується. При цьому автоматично виконуються такі дії, як компіляція, завантаження та оновлення схем баз даних, збірка на сервері. Використання CI знижує ризик появи невиявлених відразу проблем, спрощує вдосконалення програмного коду, виконує автоматизацію тривіальних дій, що дозволяє зосередитися на розробці програмної логіки підтримки функціональних можливостей системи.

4. Висновки

Запропоновані загальні підходи до створення типових рішень щодо програмної реалізації підсистем СЦ на основі ідеології і готових програмних рішень ECM-систем носять, звичайно, дискусійний характер, проте можуть, як сподіваються автори, привернути увагу фахівців і керівників, від яких залежить фінансування відповідних науково-дослідних та дослідно-конструкторських робіт, саме під цим кутом зору.

Наведена модель інтероперабельної сукупності програмних платформ забезпечує виконання всіх основних етапів по створенню підсистеми підтримки прийняття рішень – від побудови програмної інфраструктури підсистеми до модульного та інтеграційного тестування програмного коду. Слід зауважити, що однією з важливих властивостей запропонованої моделі є легке застосування систем захисту від стандартного легування до впровадження електронно-цифрових підписів. Сукупність програмно-технологічних рішень, представлених у моделі, успішно апробовано в системі підтримки прийняття рішень щодо формування і оперативної реконфігурації виробничих планів виконання договорів підприємства [19].

Загалом модель має загальний, «скелетний» характер і потребує уточнення і розвитку для конкретних застосувань. Зокрема, наведені програмні рішення базуються переважно на концепції «тонкого клієнта», що передбачає створення відповідних модулів для вирішення кожної підзадачі, з яких складаються окремі функції системи. Для деяких сукупностей задач СЦ такий підхід може виявитися нераціональним.

СПИСОК ДЖЕРЕЛ

1. Морозов А.А., Ященко В.А. Ситуационные центры – информационные технологии будущего. Київ: СП «Інтертехнодрук», 2009. 332 с.
2. Ситуационные центры и их применение. URL: http://csaa.ru/situacionnye-centry-i-ih-primenenie.
3. Морозов А.О., Кузьменко Г.Є., Литвинов В.А. Ситуаційні центри. Теорія і практика. Київ: СП «Інтертехнодрук», 2009. 348 с.
4. Гречанінов В.Ф., Кузьменко Г.Є., Лопушанський А.В., Морозов А.О. Мережа ситуаційних центрів органів державної влади – базис для підвищення ефективності їх діяльності (взаємодії). *Математичні машини і системи*. 2018. № 3. С. 32–39.
5. Макаров С. Что такое ECM. URL: https://www.osp.ru/cio/2003/04/172627.
6. Управление корпоративным контентом. URL: https://ru.wikipedia.org/wiki/Управление_корпоративным контентом.
7. Что такое ECM-системы, их преимущества в сравнении с классическими СЭД. URL: https://www.eos.ru/dop-info/chto-takoe-ECM.php.
8. Топ 10: ECM системы. URL: http://www.doc-online.ru/tools/ecm/.
9. Open-source ECM. URL: http://www.doc-online.ru/tags/open-source_ecm/.
10. Морозов А.А. Ситуационные центры. Понятия и определения. *Математичні машини і системи*. 2016. № 1. С. 48–54.
11. Tess H. The Top 13 Free and Open Source Content Management Platforms. URL: https://solutionsreview.com/content-management/the-top-free-and-open-source-content-management-platforms/.
12. Ritter S. 4 Reasons Why Java is Still #1. URL: https://www.azul.com/4-reasons-java-still-1/.
13. Walls C. Spring in Action. 5th Edition. Manning publication. 2018. 500 с.
14. Johnson R., Hoeller J. Spring Framework. Reference Documentation. URL: https://docs.spring.io/spring/docs/4.3.9.RELEASE/spring-framework-reference/html/.
15. Mihalcea M., Ebersole S. Hibernate ORM 5.2.13.Final User Guide. URL: https://docs.jboss.org/hibernate/orm/5.2/userguide/html_single/Hibernate_User_Guide.html.
16. Kalin M. Java Web Services: Up and Running. 2nd ed. O'Reilly Media, 2013. 360 с.
17. OpenAPI Specification. URL: https://swagger.io/specification/.
18. Smart J. Jenkins: The Definitive Guide. O'Reilly Media, 2011. 404 с.
19. Hrybkov S., Oliinyk H., Litvinov V. Development of information technology for supporting the process of adjustment of the food enterprise assortment. Eastern-european journal of enterprise technologies. 2018. Vol 3, N 2 (93). С. 13–24.

Стаття надійшла до редакції 12.12.2019