ISSN 2078–5097. Вісн. Львів. ун-ту. Сер. прикл. матем. та інф. 2018. Вип. 26. С. 124–136
Visnyk of the Lviv University. Series Appl. Math. and Informatics. Issue 26. P. 124–136

УДК 004.51, 004.65

РОЗРОБКА МЕНЕДЖЕРА БАЗ ДАНИХ ДЛЯ MONGODB В СЕРЕДОВИЩІ PHARO

Х. Михайлюк, В. Юріяк, С. Ярошко

Львівський національний університет імені Івана Франка,
вул. Університетська, 1, Львів, 79000, e-mail: chrismihaylyk@gmail.com

Розроблено менеджер баз даних для NoSQL MongoDB об'єктно-орієнтованою мовою програмування Smalltalk у середовищі Pharo, використовуючи шаблон MVC, бібліотеки MongoTalk та Spec. Додаток дає багато можливостей для роботи з базами даних MongoDB, а саме: створення бази, колекції або документа, зручне відображення даних, маніпулювання ними з допомогою CRUD операцій, записаних у вигляді JavaScript запитів. Великою перевагою розробленого додатку є зручність інтерфейсу та швидкість отримання, збереження й оновлення даних, що спрощує аналіз даних і роботу з ними.

Ключові слова: NoSQL бази даних, MongoDB, Smalltalk, шаблон MVC, бібліотека MongoTalk, бібліотека Spec, JavaScript запити, менеджер баз даних.

1. ВСТУП

База даних (БД) – впорядкований набір логічно взаємопов'язаних даних, організованих і структурованих за певними правилами, що передбачають загальні принципи опису, зберігання та маніпулювання цими даними. Головне призначення БД – швидкий пошук інформації, що міститься в ній. База даних вирішує такі завдання: збереження значних обсягів структурованої інформації разом з описом її структури та надання доступу до неї користувачеві або ж прикладній програмі. Програмний інтерфейс для взаємодії з БД надає система керування базами даних (СКБД).

Бази даних бувають реляційні, об'єктні, об'єктно-реляційні, документно-орієнтовані, ієрархічні, мережеві та функціональні. Найбільш вживані реляційні бази даних завдяки можливості збереження даних у вигляді цілісної структури, поданої таблицями та зв'язками між ними. Останнім часом все популярнішими стають документно-орієнтовані БД (так звані NoSQL БД). Такі тенденції простежуються тому, що виникає необхідність збереження даних, структуру яких наперед визначити важко. Для цього документно-орієнтовані БД дають змогу не лише зберігати документи різної структури в одній колекції, а й динамічно змінювати структуру одного збереженого документа, не впливаючи на інші.

В основі документно-орієнтованих СКБД є документні сховища, які мають структуру дерева. Листові вузли дерева містять дані, які при додаванні документа заносяться в індекси, що дає змогу навіть за досить складної структури знаходити місце шуканих даних чи шлях до них. Методи для пошуку допомагають знаходити за запитом документи та частини документів. Вибірка за запитом до документного сховища може містити частини великої кількості документів без повного завантаження цих документів в оперативну пам'ять, що позитивно впливає на швидкість роботи з базою даних.

Однією з NoSQL БД є MongoDB – документно-орієнтована система керування базами даних (СКБД) з відкритим вихідним кодом, яка не потребує опису схеми

© Михайлюк Х., Юріяк В., Ярошко Сергій, 2018

таблиць [1]. MongoDB оперує даними у форматі "ключ/значення", зберігає документи в JSON-подібному форматі, підтримує гнучку мову для формування запитів, може створювати індекси для різних збережених атрибутів, ефективно забезпечує зберігання великих бінарних об'єктів у колекціях.

MongoDB містить потужний сервер БД, що дає багато можливостей для роботи з базами даних, проте таку взаємодію за замовчуванням виконують лише через консоль. Сьогодні створено невелику кількість менеджерів, які надають зручний інтерфейс для роботи з MongoDB, зокрема, GUI MongoDB та крос-платформенний менеджер Robomongo.

Мета нашої праці – створити зручний у користуванні менеджер БД для сервера MongoDB, використавши мову програмування Smalltalk і можливості середовища Pharo [2]. Для розробки цього проекту доцільно використати також бібліотеки MongoTalk [3] та Spec [4]. MongoTalk надає зручні засоби для роботи з СКБД MongoDB. Для зберігання даних і отримання даних з бази не потрібно створювати окремі додаткові класи для подання колекцій в базі. Spec – це спеціальна бібліотека класів для створення різноманітних елементів управління, з допомогою яких розроблено зручний інтерфейс користувача.

2. ФОРМУЛЮВАННЯ ЗАДАЧІ

Створити додаток MongoDBBrowser, який має такі функціональні можливості:

- налагодження зв'язку з СКБД MongoDB;
- створення нових баз даних, додавання до них колекцій і документів;
- створення й оновлення колекцій і документів у вже існуючих базах даних;
- виконання основних CRUD (create, read, update, delete) – операцій з базами даних, їхніми колекціями та записами в кожній такій колекції;
- пошук по базах даних за допомогою JavaScript запитів;
- можливість швидкого опрацювання великих обсягів даних;
- зручний перегляд переліку баз даних та їхнього вмісту;
- зручний у користуванні інтерфейс.

Завдання до інтерфейсу:

- наявність зручних форм та елементів керування;
- чітке розділення основних частин вікна: області кнопок керування, області кнопок функціоналу, області відображення вмісту баз даних, області відображення документів даних;
- можливість динамічного відкривання нових вкладок для перегляду вмісту баз даних і документів у їхніх колекціях у вигляді деревовидної структури;
- відображення основної оперативної інформації для роботи з базами даних та її вмісту.

3. РЕАЛІЗАЦІЯ

Програма була спроектована з використанням шаблону MVC – модель-вигляд-контро́лер, який забезпечує чітке розподілення класів на три частини, що відповідають за внутрішню логіку програми, її візуальне зображення та взаємодію з користувачем.

3.1 МОДЕЛЬ (MODEL)

Для реалізації функціональних та інтерфейсних можливостей програмного додатку *MongoDBBrowser* було створено низку класів.

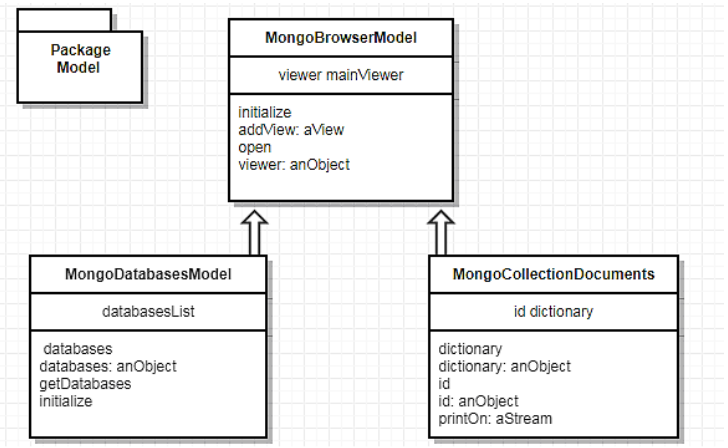

Рис. 1. Uml діаграма класів моделі для зображення даних основних структур

Усі класи, які реалізують модель, за правилами розміщення класів середовища Pharo розташовані в пакеті *Model*.

Клас *MongoBrowserModel* є базовим в ієрархії класів, які відповідають за модель даних програми. Похідними від нього є класи *MongoDatabasesModel* та *MongoCollectionDocuments* (рис. 1). Перший призначений для зображення баз даних, другий – для збереження даних моделі документів колекцій. Методи, оголошені в цих класах, застосовують для збереження даних у змінних об’єктів цих класів, а також для прив’язування об’єкта моделі до об’єкта вигляду.

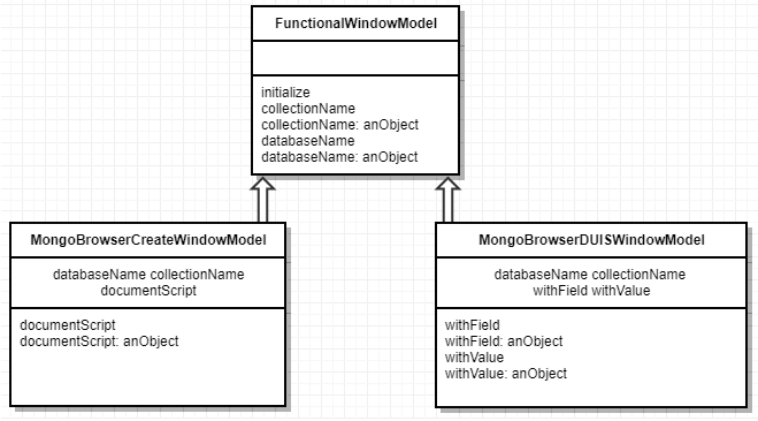

Рис. 2. Uml діаграма класів моделі для зображення даних функціональних вікон

Наступна частина класів моделі – класи для збереження даних вікон створення чи видалення баз даних: *MongoBrowserCreateWindowModel* для вікна створення бази даних та *MongoBrowserDUISWindowModel* для зображення даних автоматизованих вікон delete, update, insert та select операцій у колекціях (рис. 2).

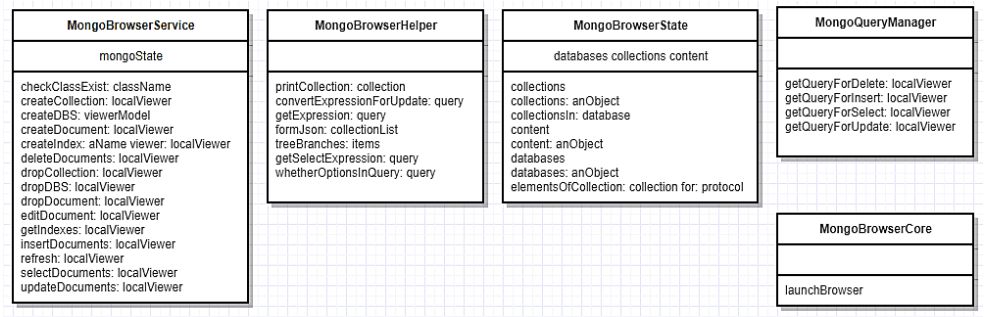

Рис. 3. Uml діаграма класів моделі для реалізації функціонала браузера

Ще одна частина класів моделі відповідає за головні функціональні операції та взаємодію з базами даних сервера MongoDB та їхнім вмістом (рис. 3). Серед них клас *MongoBrowserService*, що містить методи для створення, видалення чи оновлення баз даних, колекцій та їхніх документів.

Щоб створити базу даних, використовують метод *createDBS*. Він враховує ту особливість MongoDB, що неможливо створити базу, допоки в ній немає жодної колекції хоча б з одним документом.

Для створення колекції у певній базі даних призначено метод *createCollection: localViewer*, де його аргументом є об’єкт вигляду, що передає в метод назву вибраної в цей момент бази. Так само працює метод *createDocument*: *localViewer* для створення документа в колекції.

Для видалення бази, колекції чи документа використовують методи *dropDBS: localViewer*, *dropCollection: localViewer* та *dropDocument: localViewer*, відповідно. Перед кожним видаленням користувач отримує повідомлення про підтвердження видалення того чи іншого елемента бази, щоб запобігти випадковим небажаним діям.

Метод *updateDocument: localViewer* призначений для редагування вмісту конкретного документа. Після кожної зміни в базі даних відбувається автоматичне оновлення вмісту. Така операція забезпечується методом *refresh: localViewer*.

Клас *MongoBrowserState* відповідає за поточне збереження зміни стану баз даних та їхнього вмісту. Клас *MongoBrowserHelper* призначений для виконання допоміжних операцій. Оскільки можливості браузера для роботи з MongoDB дають змогу виконувати запити у формі JavaScript, то постає необхідність правильного перетворення цих запитів для передавання даних у методи MongoTalk. Тому в цьому класі наявні методи, серед яких *getExpression*: *query*, *convertExpressionForUpdate: query, whetherOptionsInQuery: query* тощо, за допомогою яких можна легко вибрати дані для вставлення, оновлення чи видалення документів у колекціях із JavaScript запитів. Подальше перетворення цих даних виконують методи класу *STON – Smalltalk Object Notation*, який допомагає перетворити рядкове зображення об’єкта в реальний об’єкт Smalltalk.

Клас *MongoBrowserCore* містить єдиний метод *launchBrowser* для запуску та відкривання додатку. Методи класу *MongoQueryManager* відповідають за отримання та формування запитів для основних CRUD операцій з базами даних, а саме: створення, отримання, оновлення та видалення даних у базах MongoDB.

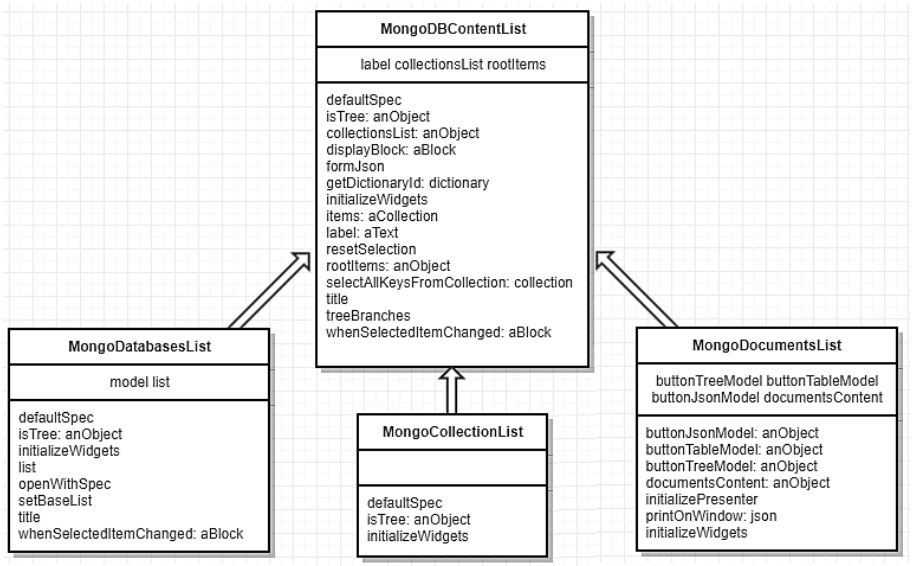

Рис. 4. Uml діаграма класів вигляду для зображення основних компонент вікна додатку

3.2 ВИГЛЯД (VIEW)

Пакет *View* містить низку класів для повноцінної реалізації можливостей зручного інтерфейсу користувача. Всі класи *View* наслідують вбудований клас Pharo Smalltalk – *ComposableModel*, який дає змогу зручно компонувати візуальні елементи.

Клас *MongoBrowserCoreView* слугує для створення оболонки основного вікна програми, панелі меню з усіма елементами керування, містить змінні для зберігання моделі програми (*browserModel*) та контролерів (*mouseControllers*, *arrowControllers*), які передаватимуть взаємодію користувача з моделлю через інтерфейс. У цьому класі методом *initializeWidgets* створюється головна панель керування програмою. Тут розташовуються різні візуальні елементи керування: кнопки, спадні списки тощо для здійснення операцій з базами даних. Також функціонал цього класу відповідає за надсилання повідомлень для відкриття допоміжних вікон програми. Наприклад, підменю для створення бази даних, колекцій і документів та для видалення кожного з цих елементів будують методи *subMenuCreate: viewer*, *subMenuDelete: viewer*, *subMenuManage: viewer*, *subMenuIndex: viewer*.

Клас *MongoBrowserViewer* відповідає за компонування основних частин вікна для відображення списків баз даних, колекцій і документів. Також він відстежує вибрані користувачем бази, колекції, документи.

Для зручного відображення вмісту колекцій і документів було реалізовано можливість відкривання вмісту колекцій або списку відфільтрованих даних після

виконання вибірок у вкладках. Для цього створено клас *TabsDynamicArea*, який містить методи *addTab: model name: aName*, *removeTab: aTab*, *tabManager* тощо. Вони допомагають керувати областю динамічних вкладок у вікні браузера.

Також для зручного виведення даних і повідомлень, щоб привертати увагу користувачів під час створення, оновлення та видалення даних, треба виділяти жирним шрифтом деякі назви. Як з'ясувалося, у Pharo Smalltalk немає засобів форматування рядка тексту, оскільки він зберігається у форматі *ByteString*. Тому довелося перевести текст для виділення жирним в тип *Text* і надіслати йому повідомлення *allBold*, всі інші "не жирні" компоненти рядка також привести до типу *Text*, після чого виконати просту конкатенацію цих підрядків.

За відображення переліку баз даних, їхніх колекцій і вмісту колекцій відповідають класи *MongoDatabasesList, MongoCollectionList* та *MongoDocumentsList*, що є похідними від класу *MongoDBContentList* (рис. 5). Перший клас забезпечує відображення переліку баз даних, дає засоби для отримання поточної виділеної бази даних. Методи інших двох класів допомагають зручно та красиво висвітлити список колекцій та їхнього вмісту. Також метод *treeBranches* допомагає відобразити вміст кожного документа в колекції у вигляді спадного списку (окреме поле в кожному окремому рядку). Для реалізації цієї можливості вміст документів зберігається у вигляді *Dictionary* та відображається за допомогою класу *TreeModel*. Коренями цього списку є дескриптори (об'єкти OID) кожного документа колекції.

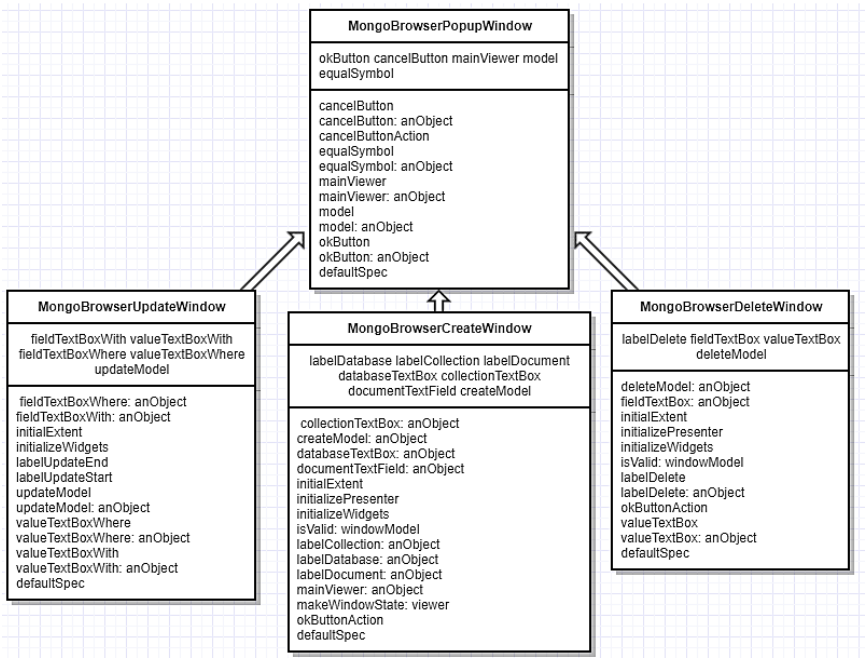

Рис. 5. Uml діаграма класів вигляду для зображення функціональних вікон

Класи, які є похідними від класу *MongoBrowserPopupWindow* – *MongoBrowser-CreateWindow, MongoBrowserDeleteWindow, MongoBrowserUpdateWindow*, призначені

для створення функціональних вікон введення інформації щодо створення, видалення чи оновлення баз даних або їхнього вмісту (рис. 5).

3.3 КОТРОЛЕР (CONTROLLER)

Ця частина програми є містком взаємодії між моделлю та виглядом.

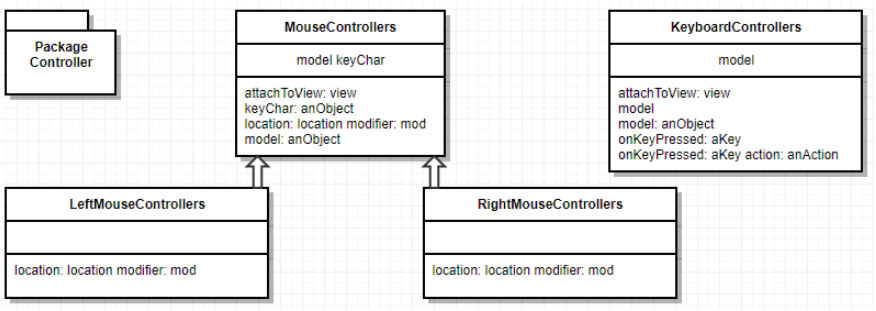

Рис. 6. Uml діаграма класів контролера

У пакеті Controller було реалізовано класи KeyboardControllers і MouseControllers, від якого наслідують два класи LeftMouseControllers та RightMouseControllers (рис. 6).

Прив’язування контролерів до моделі відбувається одразу, коли створюють об’єкти цих класів. У базовому класі реалізований метод attachToView: view, який приймає об’єкт-вигляд, і забезпечує передавання контролерів виглядам. Похідні класи реалізують метод location: location modifier: mod, який приймає місце розташування кнопки на дошці та модифікатор, що вказує, яку кнопку миші було натиснуто. Значенням модифікатора є подія yellowButtonPressed чи redButtonPressed. Для кнопок меню застосовують ліву клавішу миші, а для керування переходом по списку баз даних, колекцій і документів – обидві кнопки миші. Також надається можливість здійснення операцій за допомогою певних клавіш (див. табл.).

Список функціональних клавіш та їхніх дій

Клавіша	Дія
d	Відкриття вікна для вибору видалення або бази, або колекції, або документа
c	Відкриття вікна для вибору створення або бази, або колекції, або документа
u	Оновлення даних у документах певної колекції
r	Оновлення вмісту всіх баз даних
s	Запити до колекцій баз даних
i	Додавання нових документів
j	Виведення документів у форматі *JSON*

4. ФУНКЦІОНАЛЬНІ МОЖЛИВОСТІ ДОДАТКУ

Після аналізу роботи з консольною версією програми та деяких наявних повнофункціональних інтерфейсних програм для роботи з MongoDB було оформлено дизайн вікна додатку, реалізовано його можливості програмно, що надає користувачеві весь необхідний функціонал для роботи з MongoDB в середовищі Pharo.

MongoDBBrowser не є частиною Pharo, тому його код автори завантажили на сайт *SmalltalkHub*, і для того, щоб почати роботу з цим додатком, його треба спершу завантажити безпосередньо з сайту, за посиланням, зазначеним наприкінці цієї статті, встановити його вручну, або скористатись MonticelloBrowser для автоматичного встановлення додатку.

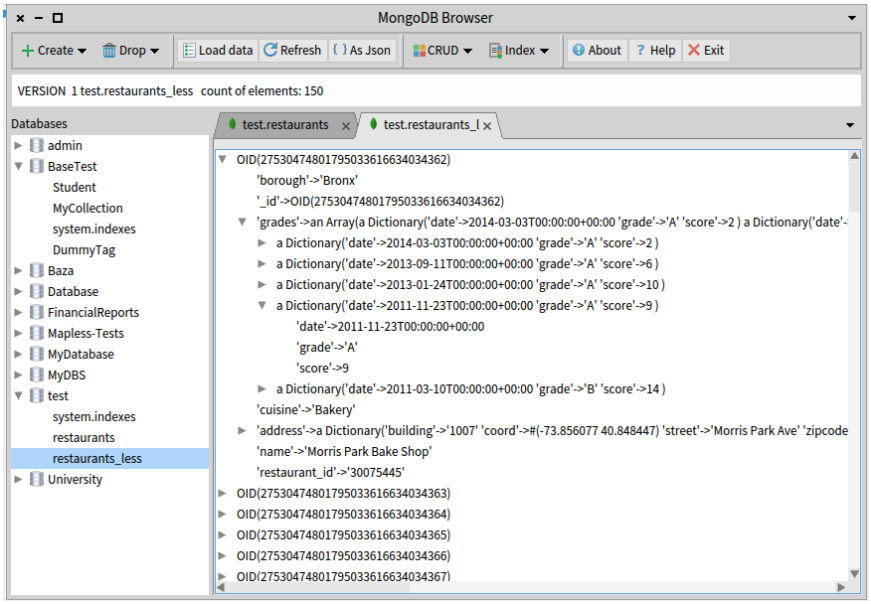

Рис. 7. Головне вікно додатку MongoDB Browser у розгорненому вигляді

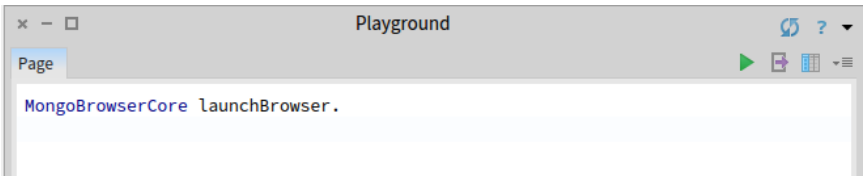

Рис. 8. Приклад запуску додатку за допомогою методу *launchBrowser*

Після завантаження та встановлення пакета *MongoDBBrowser* у головному меню в Pharo з'явиться пункт під назвою "*MongoDB Browser*", який відкриває головне вікно браузера (рис. 7). Також додаток можна запустити методом *launchBrowser* класу *MongoBrowserCore* у вікні *playground* (рис. 8).

Під кнопками головного меню додатку розташований рядок стану, де відображено інформацію про вибрану вкладку в основній частині вікна переглядача: її версія, до якої колекції належить її вміст і кількість елементів на вкладці.

На панелі ліворуч відображено список баз даних, завантажених з сервера *localhost 27017*.

Основна область вікна додатку призначена для відображення вкладок з вмістом тієї чи іншої колекції з бази даних – множини документів, результатів вибірок даних, списку індексів, інформації про програму тощо.

Зі зміною вибраної вкладки інформація рядка стану миттєво змінюється, надаючи інформацію лише щодо вибраної вкладки. Так реалізована ідея відстежування версій вкладок щодо однієї колекції. Наприклад, користувач завантажив дані з певної колекції, пізніше, запустивши скрипт, видалив частину даних і ще раз завантажив дані для відображення. У цьому випадку можна відстежити всі версії відкривання вкладок для тієї чи іншої колекції, і користувач може легко спостерігати, які дані для нього найактуальніші. Якщо вкладки попередніх версій не були закриті, то зручно можна відстежувати, скільки даних було видалено, або які саме дані були видалені тощо. Так само версії вкладок можна відстежувати, коли додають дані в колекції.

Варто зазначити, що номери версій вкладок, при закритті будь-якої з множини однотипних вкладок не зменшуються. Тобто номер версії збільшується, коли відкривається нова вкладка, та не зменшується і не занулюється, коли закривають. Відкриваючи нове вікно додатку, відлік версій починають заново.

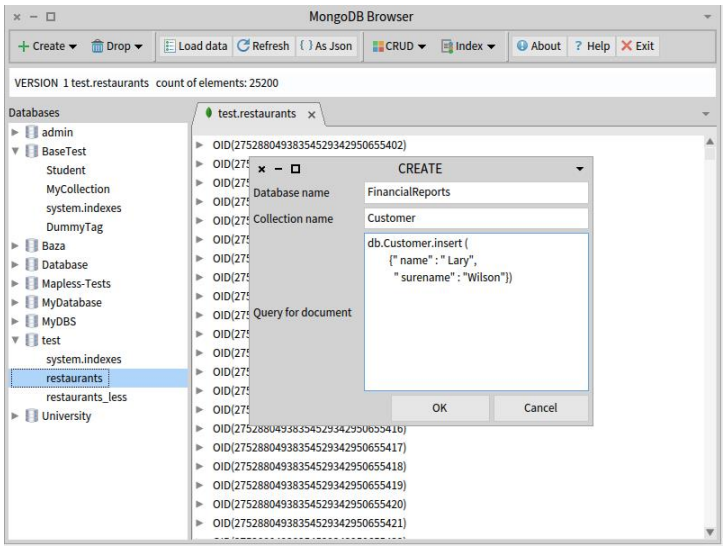

Рис. 9. Вікно створення нової бази даних

Щоб створити базу даних, користувач має вибрати відповідну підкоманду розділу меню “Create” і в спеціальному вікні діалогу (рис. 9) ввести назву нової бази даних, назву колекції та скрипт для створення щонайменше одного документа в колекції, адже відома така особливість MongoDB: якщо база даних порожня, то вона

не буде відображатись у списку баз даних. Також задля безпеки та виконання всіх цих умов вікно діалогу виконує валідацію полів: усі поля для введення є обов'язковими, якщо будь-яке з них залишиться порожнім, то буде відображено повідомлення з нагадуванням заповнити поля. Після того, як всі поля були заповнені, створена база даних відображається у списку на панелі зліва. Також користувач має змогу скасувати свої дії (створюючи базу даних чи колекцію), просто натиснувши кнопку відміни.

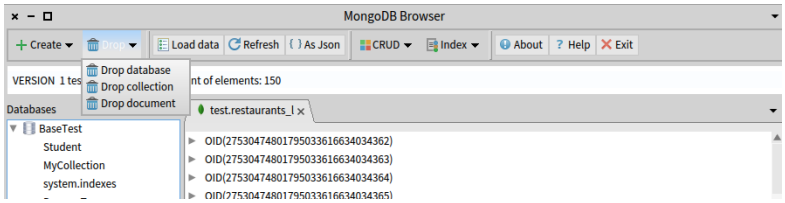

Рис. 10. Підменю видалення бази, колекції чи документа

Одразу поряд з підменю створення на панелі розташоване підменю видалення: або бази даних, або колекції, або окремого документа (рис. 10). Задля безпеки роботи з базами та їхнім вмістом при кожній спробі видалення чи то бази даних, чи колекції, чи будь-якого документа, користувачеві відображається вікно підтвердження дій, що допомагає запобігти випадковому видаленню даних.

За статистикою у користувача найчастіше виникає потреба відображати весь вміст колекції з бази даних і він змушений писати спеціальні запити до бази. Для зручності вікно *MongoDBBrowser* містить окремий елемент керування – кнопку *Load data*, яка швидко відображає весь вміст колекції в окремій вкладці. Цією ж кнопкою варто скористатися після зміни даних у базі для оновлення відображення на панелі.

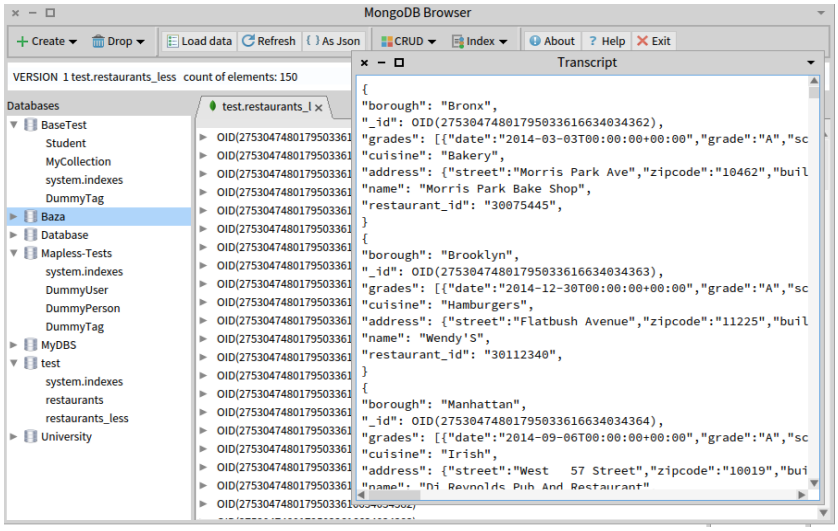

Рис. 11. JSON формат документів колекції

Для того, щоб дати користувачеві можливість скопіювати дані з певного документа чи колекції, створено опцію відображення даних у форматі *JSON*. Користувач може вибрати окремий документ або весь вміст колекції для відображення. Після натискання відповідної кнопки відкривається вікно з вмістом документів у форматі *JSON* (рис. 11), де зручно отримувати чи копіювати дані для подальшого використання.

Одне з головних завдань розробки додатку – реалізувати можливості *CRUD* операцій, які виконують за допомогою запитів мовою *JavaScript*.

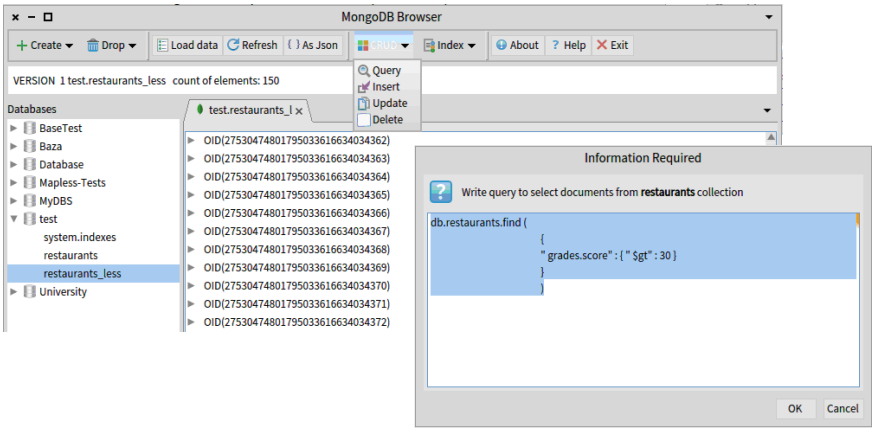

Рис. 12. Підменю для вибору CRUD операції та діалог введення скрипта для вибірки даних

Для вибору *CRUD* операції у вікні є відповідне підменю. Кожна з чотирьох наданих операцій: вибору даних, додавання документів, оновлення документів і видалення документів – відкриває діалогове вікно, яке пропонує вибір: чи відкрити автоматизоване вікно для виконання кожної з *CRUD* операцій, чи вікно з уже наявним шаблоном того чи іншого скрипта для спрощення роботи й уникнення помилок і зайвих проб при неточностях у назві колекції, команди тощо. На рис. 12 зображено приклад виконання скриптів *CRUD* операцій в *MongoDB Browser*.

Також реалізована можливість створення та перегляду індексів для кожної колекції. Під час створення індексу відкривається вікно для введення його назви.

Надані кілька службових елементів керування для відображення інформації: про програму та авторів, про те, як працювати з додатком.

5. ВИСНОВКИ

Розроблено менеджер баз даних для NoSQL MongoDB, який працює в середовищі Pharo і надає такі можливості:

- створення нової та встановлення зв’язку з наявною базою даних MongoDB;
- можливість додавання колекцій і документів;
- виконання CRUD операцій з базою даних, з її колекціями та записами в кожній такій колекції;
- пошук по базі даних за допомогою JavaScript запитів;

- швидка робота менеджера з великими об'ємами даних;
- зручний перегляд переліку баз даних та їхнього вмісту в динамічних вкладках.

Роботу менеджера перевірено на багатьох реальних базах. Програмний код додатку завантажено на сайт smalltalkhub.com, його можна знайти за посиланням http://smalltalkhub.com/#!/~KhrystynaM/MongoBDBrowser

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. MongoDB [Електронний ресурс] // Вікіпедія, 2017. – Режим доступу: https://uk.wikipedia.org/wiki/MongoDB
2. *Black A.P.* Pharo by Example / A.P. Black, S. Ducasse, O. Nierstrasz, D. Pollet, D. Cassou, M. Denker. – Kehrsatz: Square Bracket Associates, 2010. – 350 p.
3. Mapless [Електронний ресурс] / Sebastian Sastre // Github, 2017. – Режим доступу: https://github.com/sebastianconcept/Mapless
4. *Fabry J.* The Spec UI framework / J. Fabry, S. Ducasse. – Kehrsatz: Square Bracket Associates, 2017. – 78 p.

Стаття: надійшла до редколегії 12.04.2018
доопрацьовано 16.05.2018
прийнята до друку 20.06.2018

DEVELOPMENT OF THE DATABASE MANAGER FOR DBMS MONGODB IN PHARO ENVIRONMENT

K. Mykhailiuk, V. Yuriyak, S. Yaroshko

Ivan Franko National University of Lviv,
Universytetska Str., 1, Lviv, 79000, e-mail: chrismihaylyk@gmail.com

The database manager for NoSQL MongoDB has been developed using the MVC template, an object-oriented Smalltalk programming language in Pharo environment, the MongoTalk and Spec libraries. The application provides a number of features for working with MongoDB databases, such as: creating a database, collection or document, conveniently mapping and manipulating data using CRUD operations written as JavaScript requests.

At the age of fast development the information technology and increasing the amount of information exists a topical issue of choosing convenient and fast systems for saving and access to data. Now the so-called NoSQL databases are gaining popularity and rapid spread. The advantages of NoSQL databases are no secret, especially when cloud computing has gained wide adoption.

NoSQL databases were created in response to the limitations of traditional relational database technology. When compared against relational databases, NoSQL databases are more scalable and provide superior performance, and their data model addresses several shortcomings of the the relational model. One such system is MongoDB.

MongoDB is a document-oriented database management system (DBMS) with open source code that does not require a table schema description. It relates to fast and scalable systems that operate data in a key-value format. MongoDB supports the storage of documents in a JSON-like format, has a fairly flexible language for query creation, can create indexes for various stored attributes, and effectively store large binary objects in collections.

However, all operations with MongoDB databases are performed only on a console basis. Users lack access to the database in visual mode. Therefore, the purpose of our project was to create a program with a convenient window interface for managing the MongoDB databases.

The database manager for NoSQL MongoDB has been developed using the MVC template, an object-oriented Smalltalk programming language in Pharo environment, the MongoTalk, Mapless

and Spec libraries. Mapless provides convenient tools for working with the MongoDB database, automates the creation of classes for the presentation of tables in the database. Spec is a special Smalltalk class library for creating a variety of visual controls. It is easy to develop a user interface window.

The main goal of this project is to implement a desktop application with allows user to operate with MongoDB using clear and responsive interface.

The developed application works in the Pharo environment and provides the following capabilities: 1) creating a database; 2) establishing connection with the database MongoDB; 3) performing basic operations with the database – CRUD (create, read, update, delete) written as JavaScript requests - its collections and records in each such collection using ; 4) Convenient browsing of the database list and its contents.

The great advantage of the work performed is the convenience of the developed application window and the speed of obtaining, storing and updating data, which simplifies data analysis and work with them.

Key words: NoSQL Database, MongoDB, Smalltalk, MVC Template, MongoTalk Library, Spec Library, JavaScript Queries, Database Manager.