УДК 004.652.4 https://doi.org/ 10.35546/kntu2078-4481.2019.3.11

Є.В. ДАНИЛЕЦЬ
Херсонський національний технічний університет
ORCID: 0000-0003-4491-6718
Г.О. РАЙКО
Херсонський національний технічний університет
ORCID: 0000-0002-7357-5687

ВИКОРИСТАННЯ ADO.NET ENTITY FRAMEWORK ДЛЯ СТВОРЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ДОКУМЕНТООБІГОМ КАФЕДРИ

У даній роботі розглянуто використання технології ADO.NET Entity Framework для створення інформаційних систем. Показано, що архітекторам та розробникам додатків орієнтованих на роботою з базами даних, доводиться враховувати необхідність досягнення двох абсолютно різних цілей. Вони повинні модулювати сутності, зв'язки та логіку бізнес-задач, а також працювати з ядрами СУБД, що використовуються для збереження і отримання даних. Дані можуть розподілятися за кількома системами зберігання даних, в кожній з яких застосовуються свої протоколи, але навіть в додатках, що працюють з однією системою зберігання даних, необхідно підтримувати баланс між вимогами системи зберігання даних і вимогами написання ефективного і зручного для обслуговування коду програми. Для вирішення цієї проблеми існують ORM-технології програмування (Object-Relational Mapping), які зв'язують бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних». Зазначається, що використання цих технологій позбавляє програміста від написання великої кількості коду, тим самим значно підвищуючи швидкість розробки додатків. Показано, що більшість сучасних реалізацій ORM дозволяють програмісту при необхідності самому жорстко задати код SQL-запитів, який буде використовуватися при тих чи інших діях з об'єктом. В якості прикладу використання ADO.NET Entity Framework в середовищі IDE Visual Studio 2017 був створений прототип інформаційної системи управління документообігом кафедри. В роботі зазначається, що ефект від впровадження інформаційної системи кафедри полягає в інформаційній та управлінській сферах, а саме: підвищення продуктивності праці; економія часу; збільшення конкурентної переваги; забезпечення достовірності інформації; вдосконалення структури потоків інформації і системи документообігу; ефективна внутрішня координація за допомогою каналів електронного зв'язку.

Ключові слова: інформаційна система, ADO.NET Entity Framework, реляційні бази даних, SQL, Object-Relational Mapping, об'єктно-орієнтоване програмування.

Е.В. ДАНИЛЕЦ
Херсонский национальный технический университет
ORCID: 0000-0003-4491-6718
Г.А. РАЙКО
Херсонский национальный технический университет
ORCID: 0000-0002-7357-5687

ИСПОЛЬЗОВАНИЕ ADO.NET ENTITY FRAMEWORK ДЛЯ СОЗДАНИЯ ИНФОРМАЦИОННОЙ СИСТЕМЫ УПРАВЛЕНИЯ ДОКУМЕНТООБОРОТОМ КАФЕДРЫ

В данной работе рассмотрено использование технологии ADO.NET Entity Framework для создания информационных систем. Показано, что архитекторам и разработчикам приложений ориентированных на работу с базами данных, приходится учитывать необходимость достижения двух совершенно разных целей. Они должны модулировать сущности, связи и логику бизнес-задач, а также работать с ядрами СУБД, используемых для сохранения и получения данных. Данные могут распределяться по нескольким системам хранения данных, в каждой из которых применяются свои протоколы, но даже в приложениях, работающих с одной системой хранения данных, необходимо поддерживать баланс между требованиями системы хранения данных и требованиями написания эффективного и удобного для обслуживания кода программы. Для решения этой проблемы существуют ORM-технологии программирования (Object-Relational Mapping), которые связывают базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Отмечается, что использование этих технологий уменьшает объем кода, который необходимо программисту написать, тем самым значительно повышая скорость разработки

приложений. Показано, что большинство современных реализаций ORM позволяют программисту при необходимости самому жестко задать код SQL-запросов, который будет использоваться при тех или иных действиях с объектом. В качестве примера использования ADO.NET Entity Framework в среде IDE Visual Studio 2017 был создан прототип информационной системы управления документооборотом кафедры. В работе отмечается, что эффект от внедрения информационной системы кафедры заключается в информационной и управленческой сферах, а именно: повышение производительности труда; экономия времени; увеличение конкурентного преимущества; обеспечение достоверности информации; совершенствование структуры потоков информации и системы документооборота; эффективная внутренняя координация с помощью каналов электронной связи.

Ключевые слова: информационная система, ADO.NET Entity Framework, реляционные базы данных, SQL, Object-Relational Mapping, объектно-ориентированное программирование.

Y.V. DANYLETS,
Kherson National Technical University
ORCID: 0000-0003-4491-6718
H.O. RAYKO
Kherson National Technical University
ORCID: 0000-0002-7357-5687

USING ADO.NET ENTITY FRAMEWORK TO CREATE A DEPARTMENT'S INFORMATION SYSTEM

This paper discusses the use of ADO.NET Entity Framework technology to create information systems. It is shown that architects and developers of database-oriented applications have to consider the need to achieve two completely different goals. They should modulate the essence, relationships and logic of business tasks, as well as work with the database kernels used to store and retrieve data. Data can be distributed across several data storage systems, each of which has its own protocols, but even in applications that work with the same data storage system, it is necessary to maintain a balance between the requirements of the data storage system and the requirements for writing efficient and easy-to-maintain program code. To solve this problem, there are ORM-programming technologies (Object-Relational Mapping), which connect the database with the concepts of object-oriented programming languages, creating a "virtual object database". It is noted that using these technologies deprives the programmer from writing a large amount of code, thereby significantly increasing the speed of application development. It is shown that most modern ORM implementations allow the programmer to hardcode the SQL query code, if necessary, which will be used for certain actions with the object. A prototype of the department's document management information system was created in the Visual Studio 2017 IDE as an example of using the ADO.NET Entity Framework. The paper notes that the effect of the introduction of the information system of the department is in the information and management areas: improving labor productivity; time saving; increase in competitive advantage; ensuring the accuracy of information; improving the structure of information flows and the workflow system; effective internal coordination through electronic communication channels.

Keywords: information system, ADO.NET Entity Framework, relational databases, SQL, Object-Relational Mapping, object-oriented programming.

Постановка проблеми

Документи є одним з предметів праці або результатом управлінської праці, а процедури роботи з документами є невід'ємними елементами технології управління. В системі організаційного управління використовуються різні форми уявлення і використання інформації, але переважне значення має документальна форма. Документообіг будь кафедри вищого навчального закладу є одним з найбільш трудомістких процесів, що вимагає значних трудових і тимчасових витрат. При цьому важливо враховувати, що наявність в повному обсязі правильно розроблених документів є одним з найважливіших показників успішної роботи кафедри в цілому.

Як правило, кожен документ зберігається на кафедрі в двох формах: у електронному форматі (у вигляді комп'ютерних файлів на жорсткому диску комп'ютера кафедри) і в паперовому (на паперовому носії в роздрукованому вигляді, в конкретній папці документів відповідно до прийнятої класифікації). При цьому число таких папок істотно, а кількість зберігаються документів, як мінімум, на порядок більше.

Аналіз останніх досліджень і публікацій

Зазвичай архітекторам та розробникам додатків орієнтованих на роботою з базами даних, доводиться враховувати необхідність досягнення двох абсолютно різних цілей. Вони повинні модулювати сутності, зв'язки та логіку бізнес-задач, а також працювати з ядрами СУБД, що використовуються для збереження і отримання даних. Дані можуть розподілятися за кількома системами

зберігання даних, в кожній з яких застосовуються свої протоколи, але навіть в додатках, що працюють з однією системою зберігання даних, необхідно підтримувати баланс між вимогами системи зберігання даних і вимогами написання ефективного і зручного для обслуговування коду програми.

Системи керування базами даних демонструють непогану продуктивність під час виконання глобальних запитів, які зачіпають велику ділянку бази даних, але об'єктно-орієнтований доступ більш ефективний при роботі з малими обсягами даних, оскільки це дозволяє скоротити семантичну прогалину між об'єктною і реляційною формами даних [2].

В [1] зазначається, що необхідно забезпечити роботу з даними в термінах класів, а не таблиць даних і навпаки, перетворити терміни і дані класів в дані, придатні для зберігання в СУБД. Необхідно також забезпечити інтерфейс для CRUD-операцій над даними. Загалом, необхідно позбутися від необхідності писати SQL-код для взаємодії в СУБД.

Реляційні бази даних використовують набір таблиць, що представляють прості дані. Додаткова або пов'язана інформація зберігається в інших таблицях. Часто для зберігання одного об'єкта в реляційної базі даних використовується кілька таблиць; це, в свою чергу, вимагає застосування операції JOIN для отримання всієї інформації, що відноситься до об'єкту, для її обробки. Оскільки системи керування базами даних зазвичай не реалізують реляційного подання фізичного рівня зв’язків, то виконання декількох послідовних запитів може бути занадто витратним [1].

Використання реляційної бази даних для зберігання об'єктно-орієнтованих даних призводить до семантичного розриву, змушуючи програмістів писати програмне забезпечення, яке повинно вміти як обробляти дані в об'єктно-орієнтованому вигляді, так і вміти зберегти ці дані в реляційної формі. Ця постійна необхідність в перетворенні між двома різними формами даних не тільки сильно знижує продуктивність, але і створює труднощі для програмістів, так як обидві форми даних накладають обмеження друг на друга.

Для вирішення цієї проблеми існують так звані ORM (Object-Relational Mapping) - технологія програмування, яка зв'язує бази даних з концепціями об'єктно-орієнтованих мов програмування, створюючи «віртуальну об'єктну базу даних» [3].

ORM позбавляє програміста від написання великої кількості коду, тим самим значно підвищуючи швидкість розробки додатків. Крім того, більшість сучасних реалізацій ORM дозволяють програмісту при необхідності самому жорстко задати код SQL-запитів, який буде використовуватися при тих чи інших діях з об'єктом.

За даними опитування компанії Pluralsight на тему «найкраща NET ORM», перше місце за популярністю займає Entity Framework ORM, за яку віддали голоси 37,8% опитаних [4].

Формулювання мети дослідження

У багатьох вузах України впроваджуються або вже впроваджені різні інформаційні системи управління та документообігу. Найчастіше подібні системи побудовані на базі технологій найбільших зарубіжних корпорацій, таких як SAP, IBM, Microsoft, Oracle. Використання подібної технологічної платформи дозволяє значно зменшити час розробки інформаційних систем, але підвищує їх вартість. Не всі ВНЗ мають матеріальні ресурси для закупівлі подібних систем, тому часто зустрічається так звана «острівна» автоматизація, при якій автоматизуються лише окремі аспекти діяльності ВНЗ. Крім цього, існуючі системи спрямовані на автоматизацію діяльності всього вузу. Кафедра представляється у вигляді окремого модуля, який неможливо використовувати без установки всієї системи в цілому. Функції, що подаються такими системами, надлишкові, і, як наслідок, їх використання для потреб кафедри є незручним. До того ж існуючі системи вимагають значних витрат в плані матеріально-технічного забезпечення.

Все це часто відбивається на тому, що діяльність кафедри в даній області залишається не автоматизованою. У зв'язку з цим завдання розробки і впровадження інформаційних систем управління кафедрою і формування звітної документації є актуальним.

Тому метою даного дослідження є спроба використання Entity Framework ORM для вирішення нагальної потреби в інформаційній системі кафедри ВНЗ.

Викладення основного матеріалу дослідження

Entity Framework (EF) являє спеціальну об'єктно-орієнтовану технологію на базі середовища .NET для роботи з даними. Якщо традиційні засоби ADO.NET дозволяють створювати підключення, команди та інші об'єкти для взаємодії з базами даних, то EF являє собою більш високий рівень абстракції, який дозволяє абстрагуватися від самої бази даних і працювати з даними незалежно від типу сховища. Якщо на фізичному рівні ми оперуємо таблицями, індексами, первинними і зовнішніми ключами, то на концептуальному рівні, який нам пропонує EF, ми вже працюємо з об'єктами.

Середовище ADO.NET Entity Framework дозволяє додаткам взаємодіяти з даними в різних формах, включаючи дані, що зберігаються в реляційних базах даних (БД). Розробник використовує ADO.NET Entity Framework і Visual Studio для створення так званої моделі даних сутностей, що представляє базу даних, після чого за допомогою LINQ to Entities працює з об'єктами моделі сутностей.

ADO.NET Entity Framework підтримує більшість популярних систем управління базами даних, зокрема Microsoft SQL Server. EF працює таким чином, що непомітно для розробника генерує команди SQL для взаємодії з базою даних.

Центральної концепцією EF є поняття сутності або entity. Сутність представляє набір даних, асоційованих з певним об'єктом. Тому дана технологія передбачає роботу не з таблицями, а з об'єктами та їх наборами.

Будь-яка сутність, як і будь-який об'єкт з реального світу, має низку властивостей. Наприклад, якщо сутність описує співробітника кафедри, то ми можемо виділити такі властивості, як ім'я, прізвище, посада, вік тощо. Типи даних властивостей необов'язково можуть бути простими, а й можуть представляти собою більш комплексні структури даних. І у кожної сутності може бути одна або кілька властивостей, які будуть відрізняти цю сутність від інших і будуть унікально визначати цю сутність. Подібні властивості називають ключами.

При цьому сутності можуть бути пов’язані асоціативними зв'язками один-до-багатьох, один-до-одного і багато-до-багатьох, подібно до того, як в реальній базі даних відбувається зв'язок через зовнішні ключі.

Іншим важливим поняттям в EF є Entity Data Model (модель даних сутностей). Ця модель зіставляє класи сутностей з реальними таблицями в БД.

Entity Data Model складається з трьох рівнів: концептуального, рівень сховища і рівень зіставлення. На концептуальному рівні відбувається визначення класів сутностей, які використовуються в додатку. Рівень сховища визначає таблиці, стовпці, відношення між таблицями і типи даних, з якими порівнюється використовувана база даних. Рівень зіставлення служить посередником між попередніми двома, визначаючи зіставлення між властивостями класу сутності і стовпцями таблиць. Таким чином, ми можемо через класи, визначені у додатку, взаємодіяти з таблицями з бази даних.

EF передбачає три можливі способи взаємодії з базою даних: database first: EF створює набір класів, які відображають модель конкретної бази даних; model first: спочатку розробник створює модель бази даних, по якій потім EF створює реальну базу даних на сервері; code first: розробник створює клас моделі даних, які будуть зберігатися в БД, а потім EF за цією моделлю генерує базу даних та її таблиці.

Як відомо, у реляційній базі даних інформація упорядковується в таблиці. Технологія LINQ to Entities може використовуватися для визначення запитів, які обирають підмножину даних з таблиці. Відмінною рисою використання LINQ у поєднанні з EF є те, що можна отримувати не тільки окремі рядки з БД, а й цілі об’єкти, що зв’язані різними асоціативними зв’язками.

Для пояснення переваг використання описаної технології розглянемо наступний приклад, що показує як можна взаємодіяти з базою даних за допомогою платформи Entity Framework. Також в цьому прикладі продемонстровані особливості моделювання зв’язків багато-до-багатьох в ADO.NET EF.

Розглянемо спрощену базу даних УМКД, в якій зберігається інформація про елементи учбово-методичного комплексу дисципліни, схема якої побудована в Microsoft SQL Server Management Studio і зображена на рис. 1.

База даних складається з трьох таблиць: `Authors`, `ElementUMKD` і `AuthorElement`. Таблиця `Authors` складається з трьох стовпців: унікальний ідентифікатор автора, ім'я, прізвище та по-батькові.

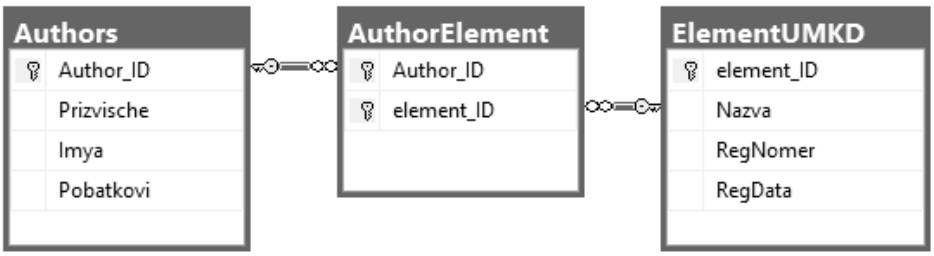

Рис. 1. Схема бази даних УМКД

Таблиця `ElementUMKD` складається з чотирьох стовпців з інформацією про кожен елемент УМКД в базі даних: в них зберігається номер елемента УМКД, назву елемента, реєстраційний номер і дату реєстрації.

Таблиця `AuthorElement` складається з двох стовпців, в яких зберігається номер елемента УМКД та ідентифікатор автора. Ця таблиця пов'язує інформацію про авторів з елементами УМКД. Стовпець `author_ID` є зовнішнім ключем, тобто значення стовпця цієї таблиці збігаються зі стовпцем первинного ключа в іншій таблиці (`Author_ID` в таблиці `ElementUMKD`). Стовпець `element_ID` теж

є зовнішнім ключем – він відповідає стовпцю первинного ключа (element_ID) з таблиці ElementUMKD.

База даних може складатися з декількох таблиць. Проектувальник БД прагне скоротити кількість даних, що дублюються між таблицями. Зовнішні ключі, що задаються в момент створення таблиці БД, пов'язують дані декількох таблиць. В даному випадку комбінація стовпців Author_ID та element_ID таблиці AuthorElement утворює складений первинний ключ. Таким чином, кожен рядок таблиці однозначно пов'язує одного автора з номером елемента УМКД.

Кожне значення зовнішнього ключа повинно відповідати значенню первинного ключа іншої таблиці, щоб СКБД могла перевірити дійсність його значення. Зовнішні ключі також дозволяють виконувати вибірку взаємопов'язаних даних з декількох таблиць – це називається об'єднанням даних. Між первинним ключем і відповідним зовнішнім ключем існує відношення «один до багатьох».

На рис. 1 первинний ключ в таблиці Authors – Author_ID, в таблиці AuthorElement – AuthorID та element_ID, в таблиці ElementUMKD - element_ID. У кожному стовпці (або групі стовпців) первинного ключа обов'язково повинно міститись значення, причому це значення повинно бути унікальним в таблиці, в іншому випадку СКБД повідомить про помилку.

Лінії, що з'єднують таблиці на рис. 1, представляють відношення між таблицями. Лінія позначає зв'язок «один до багатьох» – для кожного автора з таблиці Authors в таблиці AuthorElement може зберігатися будь-яка кількість номерів element_ID елементів УМКД, написаних цим автором (тобто автор може написати будь-яку кількість елементів УМКД).

Лінія відношення з'єднує стовпець Author_ID таблиці Authors (де Author_ID є первинним ключем) зі стовпцем Author_ID таблиці AuthorElement (де Author_ID є зовнішнім ключем) – лінія між таблицями пов'язує первинний ключ з відповідним зовнішнім ключем.

Лінія між таблицями ElementUMKD і AuthorElement представляє собою зв'язок типу «один до багатьох» – один елемент УМКД може бути написаний кількома авторами. Треба зазначити, що лінія між таблицями з'єднує первинний ключ element_ID в таблиці ElementUMKD з відповідним зовнішнім ключем таблиці AuthorElement. Схема БД на рис. 1 показує, що єдиною метою таблиці AuthorElement є створення відношення «багато до багатьох» (M:M) між таблицями Authors і ElementUMKD – автор може написати багато елементів УМКД, і у елемента УМКД може бути багато авторів.

При використанні ADO.NET Entity Framework взаємодія з базами даних відбувається через класи, які генерує IDE для схеми бази даних. Процес створення моделі даних БД в EF запускається включенням в проект нової моделі даних сутностей ADO.NET.

Для таблиць Authors і ElementUMKD в базі даних УМКД IDE створює в моделі даних два класи Authors та ElementUMKD (рис. 2), що представляють авторів елементів УМКД та самі ці елементи, а також їх властивості.

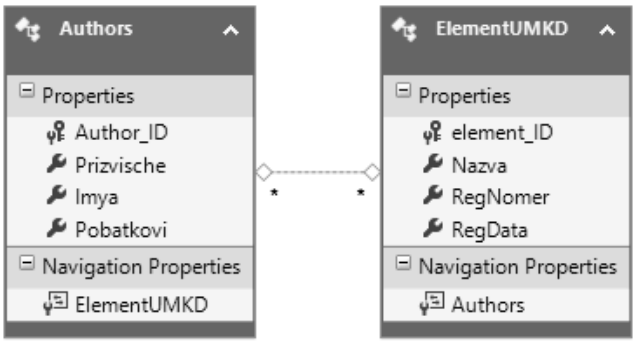

Рис. 2. Діаграма сутностей моделі даних ADO.NET Entity Framework

Важливо зазначити, що серед класів, що були згенеровані, ми не знаходимо таблицю AuthorElement. Ця таблиця пов'язує кожного автора в таблиці Authors з елементами УМКД цього автора в таблиці ElementUMKD і кожен елемент УМКД в таблиці ElementUMKD з авторами в таблиці Authors. Такі відношення між таблицями (в даному випадку - багато до багатьох або M:N) враховуються в класах моделі даних сутностей за допомогою навігаційних властивостей. Наприклад, клас Authors містить навігаційну властивість з ім'ям ElementUMKD, за допомогою якої можна отримати об'єкти ElementUMKD для всіх елементів УМКД, написаних автором. Аналогічним чином клас ElementUMKD містить навігаційну властивість Authors, за допомогою якої можна отримати

об’єкти Authors, що представляють авторів певного елемента УМКД. Фрагмент коду цих класів, що був автоматично згенерований наведений нижче:

```
public partial class ElementUMKD
{…
    public int element_ID { get; set; }
    public string Nazva { get; set; }
    public string RegNomer { get; set; }
    public Nullable<System.DateTime> RegData { get; set; }

    public virtual ICollection<Authors> Authors { get; set; }
}

public partial class Authors
{…
    public int Author_ID { get; set; }
    public string Prizvische { get; set; }
    public string Imya { get; set; }
    public string Pobatkovi { get; set; }

    public virtual ICollection<ElementUMKD> ElementUMKD { get; set; }
}
```

Навігаційні властивості дозволяють легко працювати зі зв’язаними сутнісними об'єктами, особливо коли об'єкти пов’язані зовнішнім ключем, як це показано в наступному коді:

```
using (var context = new umkd_dbEntities())
{
    var authors = context.Authors
        .Include(b => b.ElementUMKD)
        .ToList();
    foreach (var author in authors)
    {
        Debug.WriteLine(author.Imya + " " + author.Prizvische);
        foreach (var elementsUMKD in author.ElementUMKD)
        {
            Debug.WriteLine("     " + elementsUMKD.Nazva);
        }
    }
}
```

Як видно з наведеного коду, для звернення до бази даних використовується LINQ to Entities, який по суті є інтерфейсом LINQ API і, таким чином, відокремлює від фізичної бази даних сутнісну об'єктну модель даних.

В результаті маємо вивід інформації про авторів із пов’язаною інформацією про їх методичні праці у вікні Output середовища Visual Studio 2017:

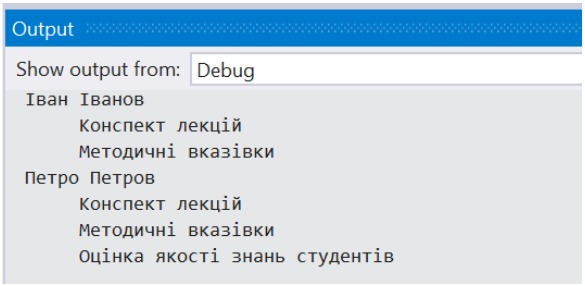

Рис. 3. Вивід запиту з БД у вікні Output

В середовищі IDE Visual Studio 2017 Community був створений прототип інформаційної системи управління документообігом кафедри з використанням платформи ADO.NET Entity Framework, головне вікно якої зображено на рис. 4.

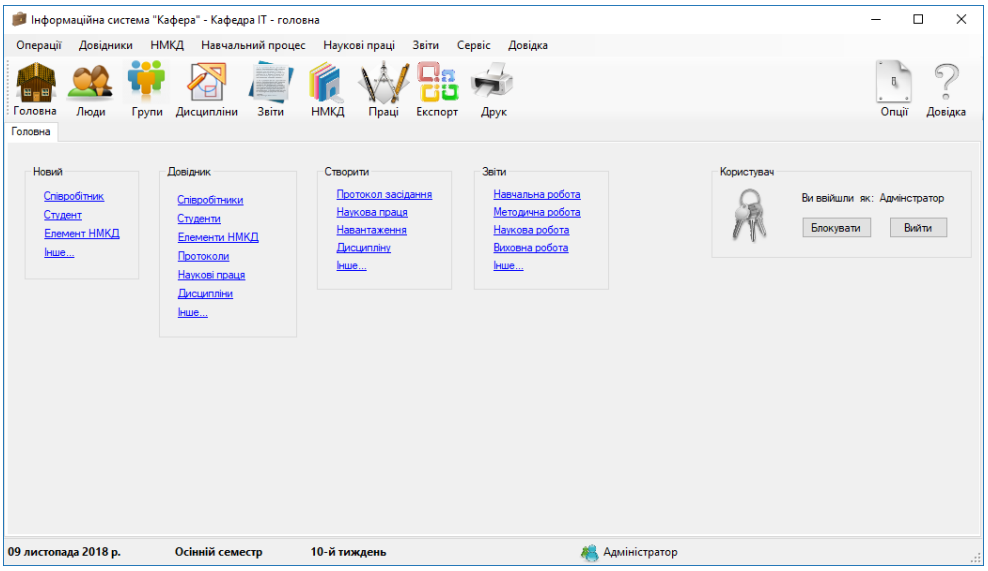

Рис. 4. Головне вікно інформаційної системи кафедри

Як вже зазначалось, створення прототипу інформаційної системи документообігу кафедри неможливе без застосування системи управління базами даних. Тому під час розробки ІС кафедри використовувалися наступні інформаційні технології та продукти: NET Framework; Microsoft SQL Server Management Studio 2016; Entity Framework v 5.0; LINQ to Entities v6; Microsoft SQL Server Express Edition 2016; об'єктно-орієнтована мова програмування C#; IDE Visual Studio Community 2017; технологія ADO.NET [3].

Основні функції інформаційної системи: збір і аналіз інформації про роботу викладачів, в тому числі публікації та участі в науково-дослідній діяльності, а також про розробку НМКД; облік інформації по дисциплінам, студентам, викладачам і аспірантам; формування звітної документації, як по календарному, так і по навчального року; розподіл навчального навантаження між викладачами, створення протоколів засідання кафедри; організація і ведення сховища даних агрегованої інформації про діяльність кафедри за минулі роки для інформаційної підтримки прийняття рішень.

Висновки

Впровадження зазначеної інформаційної системи на кафедрі підвищить ефективність її діяльності, забезпечить можливість розподіленого доступу до даних, організаційно-адміністративного супроводу інформаційної взаємодії, підвищить якість механізмів управління кафедрою, планування і контролю діяльності, управління навчальною і науково-дослідницькою діяльністю, забезпечить обмін інформацією та її облік. Крім того, система забезпечить можливість ведення і накопичення результатів діяльності та нормативних документів кафедри, моніторингу діяльності кафедри, агрегування та аналізу даних про наявний потенціал кафедри.

Ефект від впровадження інформаційної системи кафедри полягає в інформаційній та управлінській сферах, а саме: підвищення продуктивності праці; економія часу; збільшення конкурентної переваги; забезпечення достовірності інформації; вдосконалення структури потоків інформації і системи документообігу; ефективна внутрішня координація за допомогою каналів електронного зв'язку.

А використання новітніх інформаційних технологій для взаємодії з БД, зокрема ADO.NET Entity Framework, зменшує час створення інформаційної системи, зменшує кількість помилок при написанні програмного коду, спрощує налагодження і тестування системи в цілому.

Список використаної літератури

1. Дейтел П. Как программировать на Visual C# 2012. 5-е изд. / П. Дейтел, Х. Дейтел – СПб.: Питер, 2014. – 864 с.
2. Фримен Адам LINQ: язык интегрированных запросов в C# 2010 для профессионалов.: Пер. с англ. / Адам Фримен, Джозеф С Раттц-мл. – М.: ООО "И.Д. Вильямс", 2011. – 656 с.
3. Ноубл Дж., Flex 4. Рецепты программирования / Дж. Ноубл, Т. Андерсон, Г. Брэйтуэйт, М. Казарио, Р. Третола – БХВ-Петербург, 2011. – 720 с.
4. Данилець Є.В Моделювання зв'язків m:m баз даних в ADO.NET Entity Framework / Є.В. Данилець, О.О. Панфіленко // Молодь у світі сучасних технологій: Матеріали VI Міжнародної науково-практичної конференції студентів, аспірантів та молодих вчених. – Херсон: ХНТУ. – 2017. – С. 115 – 117.

References

1. Deytel P., Deytel Kh. Visual C# 2012 How to Program Pearson Education, Inc., 2013. 1024 p. (Rus. ed.: Deytel P., Deytel Kh. Kak programmirovat' na Visual C# 2012. SPb.: Piter, 2014. 864 p.).
2. Adam Frimen, Joseph C. Rattz, Jr. Pro LINQ Language Integrated Query in C# 2010 Apress. 2010. 725 p. (Rus. ed.: Adam Frimen LINQ: yazyk integrirovannykh zaprosov v C # 2010 dlya professionalov. Moscow: Williams Publishing House, 2011. 656 p.).
3. Joshua Noble, Todd Anderson, Garth Braithwaite,Marco Casario, and Rich Tretola. Flex 4 Cookbook. O'Reilly Media. 2010. 764 p. (Rus. ed.: Joshua Noble, Todd Anderson, Garth Braithwaite,Marco Casario, and Rich Tretola. Flex 4. Retsepty programmirovaniya. BKHV Peterburg, 2011. 720 p.
4. Danilets Y.V., Panfilenko O.O. Modeling of m: m database connections in ADO.NET Entity Framework. Anotatsiï dopovidei VI Mizhnarodnoyi naukovo-praktychnoyi konferentsiyi studentiv, aspirantiv ta molodykh vchenykh "Molod′ u sviti suchasnykh tekhnolohiy" [Abstracts of VI International Scientific and Practical Conference of Students, Graduate Students and Young Scientists "Youth in the World of Modern Technologies"]. – Kherson, 2017, p. 115 - 117.