УДК 004.4'24 https://doi.org/10.15407/pp2021.01.003

А.Ю. Дорошенко, Б.В. Бодак

МОДЕЛЮВАННЯ RESTFUL API ДЛЯ СИСТЕМИ АВТОМАТИЗАЦІЇ ПРИВАТНИХ ЕЛЕКТРОННИХ ЗАКУПІВЕЛЬ

Розроблено програмний засіб для автоматизації електронних закупівель на основі .NET Core RESTful API з використанням специфікацій OpenAPI v3.0. Реалізовано авторизацію користувачів системи, постачальників та замовників, за допомогою відкритого стандарту OAuth та Microsoft Identity Server. Для скорочення часу відгуку системи здійснено кешування даних на рівні репозиторію за підтримки розподіленного кешу. Створено підсистему для обробки та переходу між станами закупівель на основі скінченного автомату станів. Проведено випробування розробленого програмного засобу з використанням модульних та інтеграційних тестів.

Ключові слова: електронні закупівлі, RESTful API, OpenAPI v3.0, архітектура API, масштабовані системи автоматизації.

Вступ

Автоматизація процесів у системах закупівель набула широкого використання в багатьох країнах Європи та світу [1]. Як свідчить досвід європейських країн створення системи електронних торгів дозволяє суттєво зменшити витрати державних коштів замовниками торгів на придбання предметів закупівлі, організацію та проведення закупівельного процесу; дає змогу безперешкодного, щодобового он-лайн доступу до інформації всіх зацікавлених осіб; забезпечує відкритість та прозорість на всіх стадіях закупівель; скорочує паперовий документообіг у цій сфері; створює рівний доступ до накопичуваної інформації та рівні умови конкуренції між постачальниками; полегшує роботу контролюючих та правоохоронних органів; попереджує зловживання та корупційні прояви у сфері закупівель тощо. Зазвичай, подібні системи автоматизації тендерних закупівель складаються з декількох модулів, серед яких можна виділити кабінет замовника, постачальника, та модуль аукціону. Мета даної роботи – створення серверної частини системи автоматизації електронних закупівель для приватного бізнесу.

Основний фактор у системі електронних закупівель – аукціон, а точніше його зворотня форма. Подібний тип прийнято називати «редукціоном», оскільки це аукціон, у якому перемагає найбільш економічно вигідна пропозиція. Згідно сайту Business Dictionary, зворотній аукціон – це «Тип аукціону, у якому декілька постачальників пропонують свої товари та змагаються за ціну, яка буде прийнятою для замовника. Замовник зазвичай має можливість прийняти будь-яку пропозицію або відхилити всі пропозиції» [2]. Наявність даного типу аукціону головний критерій при аналізі існуючих рішень на ринку.

Станом на даний момент існує декілька систем, які пропонують готові рішення для веб-сайтів пов’язаних з аукціоном. На прикладі існуючих систем PHP Pro Bid [3] та iLance [4], ми порівняємо розроблений модуль аукціону та виділим переваги нашої системи автоматизації для електронних закупівель. PHP Pro Bid – це комплексна система, яка пропонує повноцінний функціонал для роботи з користувачами, аукціонами та різними методами оплати товарів чи послуг. При створенні аукціону користувачу надається можливість вказувати деталі, такі як назва товару, опис, ціна, зображення, мінімальна ставка та інше. Також аукціон автоматично продовжується якщо ставку зроблено в останній момент перед закриттям. Створення нового аукціону відбувається через панель адміністратора, що являється недоліком, оскільки у електронних закупівлях аукціони створюються замовником. У PHP Pro Bid не існує функціоналу для реєстрації користувачів різних ролей – замовник та

постачальник, що було реалізовано у нашій системі автоматизації електронних закупівель. Виділення ролей користувачів з певним функціоналом, створення єдиного сервісу для авторизації та реєстрації, дозволило зробити нашу систему масштабованою.

У свою чергу система iLance включає у себе різновидність модулів для автоматизації аукціонів. За допомогою шаблонів та широкого обсягу налаштувань у системі можливо створювати аукціони для різних видів товарів та послуг. Однак у даному фреймворку відсутня можливість оголошувати аукціон відкритого типу, що безумовно є ключовим у електронних закупівлях. На відміну від закритого аукціону, спостерігачам нашої платформи надається можливість переглядати інформацію за пропозиціями до закупівлі та постачальниками. Таким чином ми забезпечуємо найбільш можливу прозорість закупівлі та відкриту конкуренцію.

Аналіз різновидностей аукціонів показує що існує декілька типів зворотнього відкритого аукціону, а саме: Голанський, Японський, Американський та Англійський [5]. Останній тип – найбільш поширений у системах електронних закупівель, тому в нашій статті ми зосередимося на проектуванні та реалізації даного виду аукціону за допомогою кінцевого автомату станів Stateless [6]. Зворотній Англійський аукціон дозволяє постачальникам напряму взаємодіяти із замовником. Такий аукціон припиняється коли залишається лише один постачальник або час проведення завершується. Нарешті перемагає пропозиція з найбільш вигідною ціною для замовника, але замовник також матиме можливість обрати переможця за іншими нецiновими критеріями.

На сервері також використовується стандарт RESTFul API [7] забезпечує сумісність системи з більшістю сучасних клієнтів та дозволяє працювати з сутностями через обмін моделей даних у форматі JSON.

1. Формування вимог до системи

Система має надавати можливість замовнику оголошувати закупівлі на певні товари або послуги, запрошувати до участі постачальників. Замовник зможе створювати лоти та позиції товарів, критерії для відбору пропозицій (цінові та нецінові). До кожної позиції замовник може вказувати галузь закупівлі та місце поставки товарів.

У свою чергу, система також повинна автоматично запрошувати перевірених постачальників у галузі, якщо користувач обрав таку можливість. Електронний документообіг та укладання договорів через електронний цифровий підпис також мають бути передбачені у системі. Замовники зможуть завантажувати документи до закупівлі, а також переглядати документи завантажені постачальником.

Замовники та постачальники зможуть спілкуватися та обговорювати умови закупівлі через систему обговорень. Закупівлі мають проводитися відкрито, тобто будь-хто зі спостерігачів матиме можливість зайти до системи та переглянути активні тендери.

Діаграми використання системи для таких ролей користувача, як замовник, постачальник, та спостерігач (не авторизований користувач) показано на рис. 1–3.

Розроблена діаграма використання відображає базовий набір функцій, які доступні не авторизованому користувачу системи автоматизації електронних закупівель, а саме спостерігачу.

Оскільки система призначена для зареєстрованих користувачів, то функціонал спостерігача обмежено переглядом існуючих закупівель, пошуку тендера за номером, ключовими словами або назвою. Таким чином, спостерігач має доступ до перегляду усіх наявних у системі закупівель, які він може фільтрувати за такими критеріями як галузь, бюджет, ЄДРПОУ замовника, статус, дата початку та завершення.

У спостерігача наявна можливість зареєструватися у системі як замовник для подальшої роботи та отримання повноцінного функціоналу.

Система орієнтована на потреби замовника в процесі автоматизації електронних закупівель, та надає такі можливості, як оголошення закупівлі, перегляд власних закупівель, перегляд пропозицій, обговорення та обмін документами, вибір переможця та підписання договору.

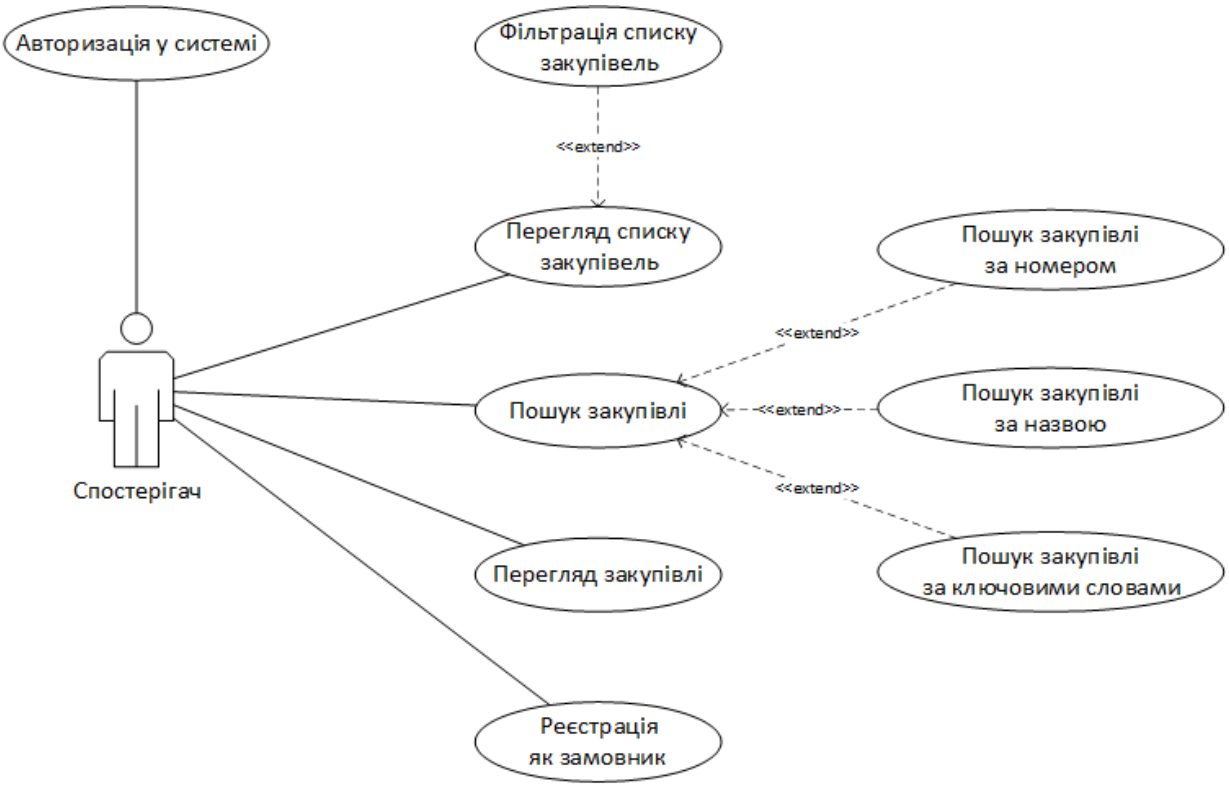

Рис. 1. Діаграма використання спостерігачем

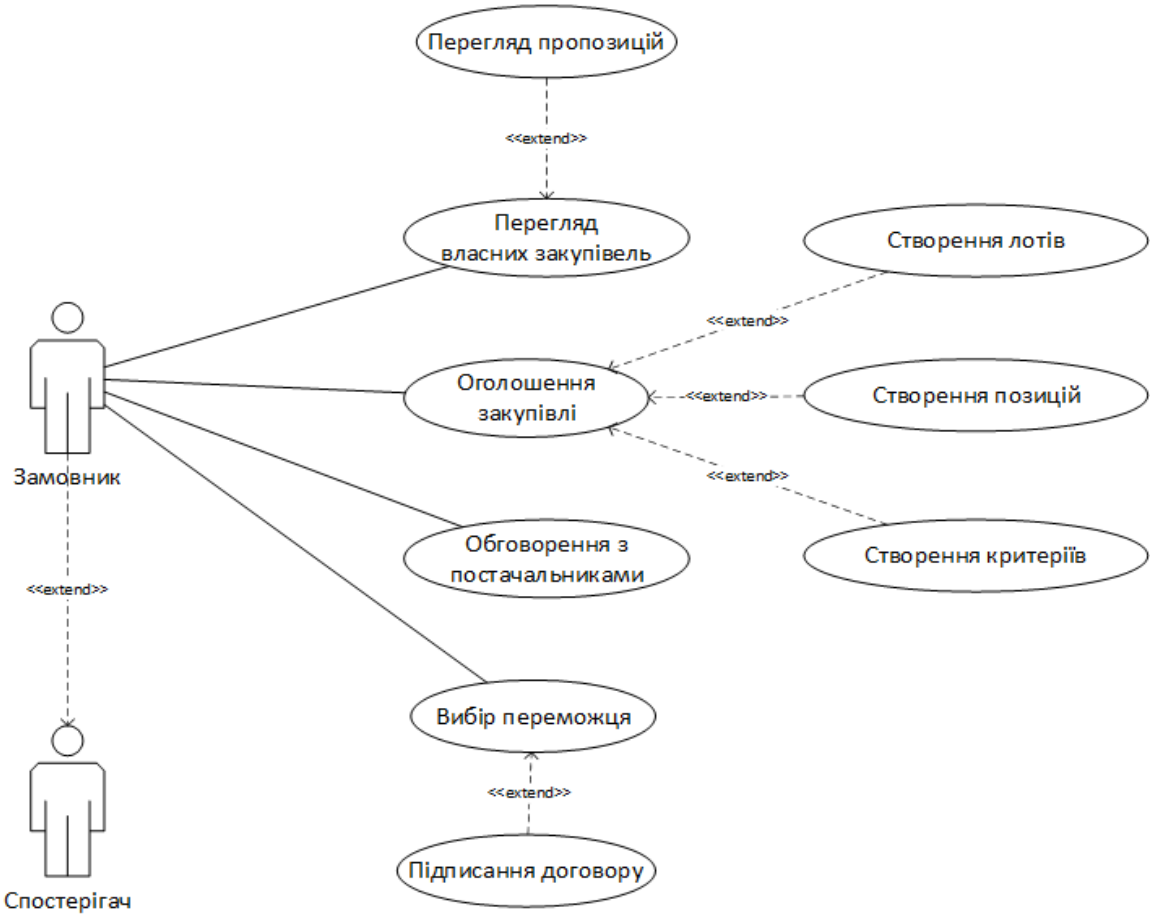

Рис. 2. Діаграма використання замовником

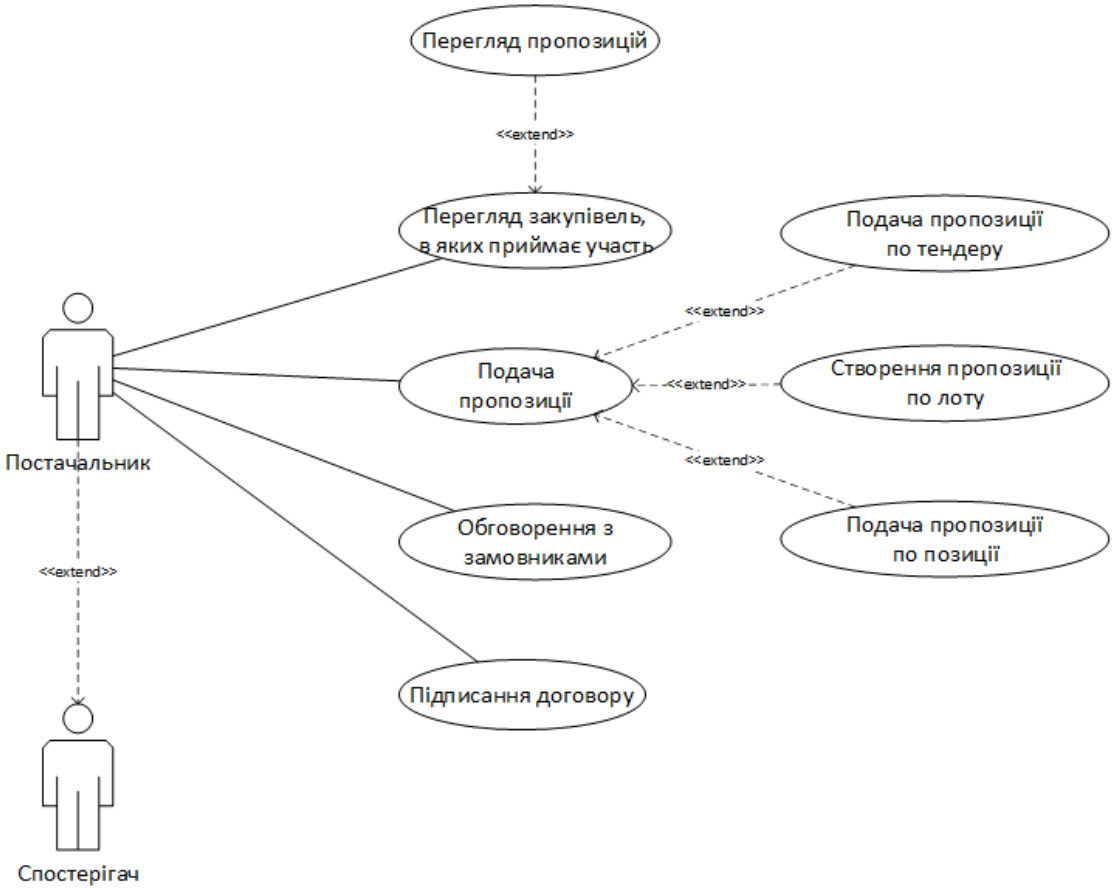

Рис. 3. Діаграма використання постачальником

Постачальник включає всі можливості спостерігача у системі та доповнює їх можливістю подавати пропозиції до закупівель, лоті та позицій, переглядати подані пропозиції, спілкуватися з замовником та підписувати контракт.

Після формування варіантів використання системи для різних ролей користувачів, було виділено основні сутності та атрибути для проектування рівня доступу до даних. Перелік сутностей та атрибутів наведено у табл. 1.

Таблиця 1. Сутності та атрибути

Сутність	Атрибути
Тендери	Статус, Назва, Опис, Бюджет, Валюта, ПДВ, Початок прийому пропозицій, Кінець прийому пропозицій, Інформація про компанію, Контактна особа, Тип закупівлі, Додаткові атрибути, Список нецінових показників, Список документів, Список критеріїв, Дата створення, Дата модифікації, Видалений
Лоти	Тендер, Етап, Назва, Опис, ПДВ, Статус, Додаткові атрибути, Список нецінових показників, Список документів, Список критеріїв, Дата створення, Дата модифікації, Видалений
Пропозиції	Тендер, Лот, Статус, Постачальник, Email, ЕДРПОУ, Назва компанії, Додаткові параметри, Видалена
Контракти	Пропозиція, Постачальник, Статус, Сума
Позиції	Лот, Назва, Опис, Кількість, Одиниці виміру, Додаткові атрибути, Список нецінових показників, Список документів, Список критеріїв, Дата створення, Дата модифікації, Виделено
Етапи	Номер етапу, JSON тендера

Проектування рівня доступу до даних починається з побудови логічної моделі БД. Незалежно від моделі даних, побудова логічної моделі БД на практиці виконується з урахуванням двох основних вимог: виключити надмірність і максимально підвищити якість даних. Ці вимоги випливають із вимоги колективного використання даних групою користувачів. На даному етапі потрібно перетворити об'єкти та зв'язки між ними у логічну модель даних – реляційну модель. В ній об'єкти та зв'язки між ними представлені у вигляді таблиць (відношень), які складаються із стовпців та рядків. Кожне поле має ім'я та тип. Тип задає спосіб представлення поля. Також визначимо для атрибутів чи обов'язково вони повинні містити дані, чи допускаються нульові значення. Для системи автоматизації електронних закупівель визначені наступні поля, значення, типи даних та обмеження, які наведені у таблицях 2–5.

Таблиця 2. Фізична модель закупівлі

Назва поля	Значення	Тип даних	Обмеження
1	2	3	4
Id	Ідентифікатор	BIGINT	PK, NN
Guid	Унікальний ідентифікатор	UNIQUEIDENTIFIER	NN
Status	Статус	INT	NN
Title	Назва закупівлі	NVARCHAR(1500)	NN
Description	Опис закупівлі	NVARCHAR(2500)	-
Budget	Бюджет закупівлі	MONEY DEFAULT (0)	-
CurrencyIso	Код валюти у форматі ISO	INT	NN
IsVat	Закупівля з ПДВ	BIT DEFAULT (0)	-
StartBid	Дата та час початку прийому пропозицій	DATETIMEOFFSET(7)	-
EndBid	Дата та час кінця прийому пропозицій	DATETIMEOFFSET(7)	-
InfoCompany	JSON інформації про компанію	NVARCHAR(MAX)	NN
ContactPerson	JSON інформації про контактну особу	NVARCHAR(MAX)	NN
TenderType	Тип закупівлі: аукціон, редукціон, запит цінових пропозицій	INT	NN
AdditionalAttributes	JSON будь-яких додаткових властивостей закупівлі (для підтримки нових підключень)	NVARCHAR(MAX)	-
ListFeatures	JSON списку нецінових показників	NVARCHAR(MAX)	-
ListDocuments	JSON списку документів до закупівлі	NVARCHAR(MAX)	-
ListCriteriaGroups	JSON критеріїв закупівлі, встановлених замовником	NVARCHAR(MAX)	-
DateCreate	Дата оголошення закупівлі	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	NN
DateModified	Дата внесення змін до закупівлі	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	-
IsRemoved	Чи було видалено закупівлю із системи	BIT DEFAULT (0)	-

Таблиця 3. Фізична модель лотів закупівлі

Назва поля	Значення	Тип даних	Обмеження
1	2	3	4
Id	Ідентифікатор	BIGINT	PK, NN
TenderId	Ідентифікатор пов'язаного тендера	BIGINT	FK, NN
Guid	Унікальний ідентифікатор	UNIQUEIDENTIFIER	NN
Status	Статус	INT	NN
Title	Назва лоту	NVARCHAR(1500)	NN
Description	Опис лоту	NVARCHAR(2500)	-
IsVat	Закупівля з ПДВ	BIT DEFAULT (0)	-
AdditionalAttributes	JSON будь-яких додаткових властивостей лота (для підтримки нових підключень)	NVARCHAR(MAX)	-
ListFeatures	JSON списку нецінових показників	NVARCHAR(MAX)	-
ListDocuments	JSON списку документів до лота	NVARCHAR(MAX)	-
ListCriteriaGroups	JSON критеріїв лота, встановлених замовником	NVARCHAR(MAX)	-
DateCreate	Дата створення лота	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	NN
DateModified	Дата внесення змін до лота	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	-
IsRemoved	Чи було видалено лот із системи	BIT DEFAULT (0)	-

Таблиця 4. Фізична модель пропозицій до закупівлі

Назва поля	Значення	Тип даних	Обмеження
1	2	3	4
Id	Ідентифікатор	BIGINT	PK, NN
StageId	Ідентифікатор пов'язаного етапу	BIGINT	FK, NN
TenderId	Ідентифікатор пов'язаного тендеру	BIGINT	FK, NN
LotId	Ідентифікатор пов'язаного лоту	BIGINT	FK, NN
Guid	Унікальний ідентифікатор	UNIQUEIDENTIFIER	NN
Status	Статус пропозиції	INT	NN
UserId	Ідентифікатор постачальника	BIGINT	NN
Email	Email постачальника	NVARCHAR(50)	NN
Edrpou	Код постачальника у реєстрі	NVARCHAR(15)	NN
CompanyName	Назва компанії постачальника	NVARCHAR(50)	NN
BidParameters	JSON списку додаткових параметрів закупівлі	NVARCHAR(MAX)	-

1	2	3	4
DateCreate	Дата створення пропозиції	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	NN
DateModified	Дата внесення змін до пропозиції	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	-
IsRemoved	Чи було видалено пропозицію із системи	BIT DEFAULT (0)	-

Таблиця 5. Фізична модель контрактів для закупівлі

Назва поля	Значення	Тип даних	Обмеження
Id	Ідентифікатор	BIGINT	PK, NN
StageId	Ідентифікатор пов'язаного етапу	BIGINT	FK, NN
TenderId	Ідентифікатор пов'язаного тендеру	BIGINT	FK, NN
LotId	Ідентифікатор пов'язаного лоту	BIGINT	FK, NN
ItemId	Ідентифікатор пов'язаної позиції	BIGINT	FK, NN
BidId	Ідентифікатор пов'язаної пропозиції	BIGINT	FK, NN
UserId	Ідентифікатор постачальника	BIGINT	FK, NN
Status	Статус договору	INT	NN
Rate	Сума договору	DECIMAL	NN

Додатково були розроблені збережені процедури для створення та оновлення пропозицій, отримання списку пропозицій по закупівлі, лоту, позиції, створення та редагування контрактів, отримання контрактів закупівлі. Первинні та зовнішні ключі забезпечують цілісність даних у БД. У таблицях 6 та 7 представлено фізичну модель товарів та етапів закупівлі відповідно.

Таблиця 6. Фізична модель товарів для закупівлі

Назва поля	Значення	Тип даних	Обмеження
1	2	3	4
Id	Ідентифікатор	BIGINT	PK, NN
LotId	Ідентифікатор пов'язаного лоту	BIGINT	FK, NN
Guid	Унікальний ідентифікатор	UNIQUEIDENTIFIER	NN
Title	Назва позиції	NVARCHAR(1500)	NN
Description	Опис позиції	NVARCHAR(2500)	-
Quantity	Кількість товарів у позиції	FLOAT DEFAULT (0)	NN
UnitId	Ідентифікатор одиниць вимірювання	INT	NN
AdditionalAttributes	JSON будь-яких додаткових властивостей позиції (для підтримки нових підключень)	NVARCHAR(MAX)	-
ListFeatures	JSON списку нецінових показників	NVARCHAR(MAX)	-
ListDocuments	JSON списку документів до позиції	NVARCHAR(MAX)	-

1	2	3	4
ListCriteriaGroups	JSON критеріїв лота, встановлених замовником	NVARCHAR(MAX)	-
DateCreate	Дата створення позиції	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	NN
DateModified	Дата внесення змін до позиції	DATETIMEOFFSET(7) DEFAULT (SYSDATETIMEOFFSET())	-
IsRemoved	Чи було видалено позицію із системи	BIT DEFAULT (0)	-

Таблиця 7. Фізична модель етапів закупівлі

Назва поля	Значення	Тип даних	Обмеження
Id	Ідентифікатор	BIGINT	PK, NN
StageNumber	Номер етапу закупівлі	INT	NN
TenderId	Ідентифікатор пов’язаного тендеру	BIGINT	FK, NN
TenderJson	JSON інформації про закупівлю на минулому етапі	NVARCHAR(MAX)	-

2. Реалізація бізнес-логіки

У системі бізнес-логіка зосереджена в мікросервісах, які спілкуються з клієнтом у форматі обміну даних JSON. Мікросервіси викликаються через контролери-маршрутизатори, що робить її закритими від зовнішнього світу. Крім цього при запитах на модифікацію даних передаються відповідні заголовки авторизації для уникнення несанкціонованого доступу.

Сервіс TenderService призначено для роботи з закупівлями, наприклад, створення нової закупівлі, отримання існуючої закупівлі, отримання документів, видалення закупівлі. Клас сервісу складається з таких методів:

- UpsertTender. Метод виконує логіку створення нової закупівлі або оновлення вже існуючої. Закупівлю може створити лише замовник, який зареєстрований у системі та авторизований. Вхідним параметром є об’єкт типу TenderDTO, який містить дані для коректного створення закупівлі. Валідація параметру проводиться на контролері, тому сервіс лише викликає збережену процедуру для створення або оновлення закупівлі;
- DeleteTender. Метод виконує логіку видалення закупівлі за ідентифікатором. Для забезпечення можливості перегляду архівних закупівель, дані із бази не видаляються, замість цього у базі даних за допомогою збереженої процедури встановлюється поле isRemove для закупівлі у значення true;
- GetTenderById. Метод повертає закупівлю за ідентифікатором. Викликається збережена процедура, яка збирає із бази даних інформацію по тендеру та повертає результат у форматі JSON. Після чого метод за допомогою конверторів створює модель закупівлі TenderDTO та повертає її на контролер, який потім повертає цю модель клієнту;
- GetTenderDocumentsById. Метод містить логіку для отримання списку документів до закупівлі. Викликає збережену процедуру, що повертає документи за типом об’єкта (тендер, лот, позиція) з параметром тендер. Результат збереженої процедури оброблюється через конвертор і створюється список DocumentDTO, який містить інформацію по кожному документу: назва, тип, посилання, дата створення. Список повертається контролеру, який повертає його клієнтові.

За аналогічною схемою реалізовано класи для роботи з лотами (LotService) та позиціями (ItemService) закупівлі.

Патерн Dependency Injection [8] дуже широко використовується в даному проекті. Як ціль було прийнято відійти від

використання конкретними класами, а користуватися лише інтерфейсами, зменшити використання ключового слова new та полегшити тестування проекту. Для цих цілей використано стандартний механізм DI в .NET Core. Вибір обгрунтовується зрозумілим API, розгорнутою та повною документацією. В даному проекті використано стандартні механізми фреймворку .NET Core для реєстрації контролерів та підтримки режиму часу життя сутностей на протязі одного запиту до WebApi. Це дає нам можливість розробляти бізнес-логіку не звертаючи увагу на час життя сутності, тому що він визначається в реєстрації сутності.

Сервіс StateService призначено для роботи з статусами та етапами закупівель, переведення закупівлі на новий етап, зміну статуса. Клас складається з наступних методів:

- GetStateByTenderId. Метод містить логіку для отримання статуса закупівлі за ідентифікатором. Викликає відповідну збережену процедуру, яка знаходить закупівлю у БД і повертає її статус;
- ChangeStateTender. Метод виконує логіку по зміні статуса тендера на вказаний. Викликає збережену процедуру, яка змінює статус для закупівлі у БД на вказаний;
- MoveToNextStage. Метод переводить закупівлю за ідентифікатором на наступний етап (наприклад на підписання контракту). Викликається збережена процедура, яка перевіряє поточний етап тендера та змінює його на наступний.

Для реалізації станів закупівлі та переходу між ними використано фреймворк Stateless та Hangfire. Клас StateService використовується машиною станів TenderStateMachine. Спочатку створено базовий клас для кінцевого автомату BaseStateMachine:

```
internal abstract class BaseStateMachine: BaseService
  {
    public readonly IStateService _stateService;
    public BaseStateMachine(IStateService
stateService,IPublishService
publishService):base(publishService)
    {
      _stateService = stateService;
    }
    public abstract object ChangeState(object model);
  }
```

За допомогою механізму наслідування було створено реалізацію:

```
internal class TenderStateMachine : BaseStateMachine
  {
…
    // <summary>
    /// Configuration state
    /// </summary>
    private void ConfigureStateMachine()
    {
      _machine = new
StateMachine<TenderStateMachineStatus,
TenderStateMachineTrigger>(() =>
_TenderStateMachineStatus, s =>
_TenderStateMachineStatus = s);
      _triggerParams =
_machine.SetTriggerParameters<TenderSync>(Tender
StateMachineTrigger.ChangeStatus);


_machine.Configure(TenderStateMachineStatus.New)

.PermitIf(TenderStateMachineTrigger.ChangeStatus,
TenderStateMachineStatus.Planned);


_machine.Configure(TenderStateMachineStatus.Planne
d)

.Permit(TenderStateMachineTrigger.ChangeStatus,
TenderStateMachineStatus.AcceptanceOfOffers)
      .OnEntryFrom(_triggerParams, model =>
AcceptanceOfOffers(model));


_machine.Configure(TenderStateMachineStatus.Accep
tanceOfOffers)

.PermitTenderStateMachineTrigger.ChangeStatus,
TenderStateMachineStatus.DataProcessing)
      .OnEntryFrom(_triggerParams, model =>
AcceptanceOfOffers(model));


_machine.Configure(TenderStateMachineStatus.DataP
rocessing)

.Permit(TenderStateMachineTrigger.ChangeStatus,
TenderStateMachineStatus.Completed)
      .OnEntryFrom(_triggerParams, model =>
DataProcessing(model));
```

```
_machine.Configure(TenderStateMachineStatus.Compl
eted)
            .OnEntryFrom(_triggerParams, model =>
Task.Run(async () => await PushAwards(model)))
            .OnEntryFrom(_triggerParams, model =>
Task.Run(async () => await UpdateState(model)));

        }

        /// <summary>
        /// Changed Status
        /// </summary>
        /// <param name="state"></param>
        private async Task UpdateState(Tender model)
        {
            model.Status = (int)_machine.State;
            var result = await
_stateService.InsertOrUpdate(new StateTender()
            {
                TenderId = model.Id,
                StageNumber = model.StageNumber,
                StatusId = (int)_machine.State,
                JobId = _currentTenderState?.JobId,
                TenderJson =
JsonConvert.SerializeObject(model)
            });

            await
_stateService.UpdateStatusTender(model);
        }
…
}
```

Стани закупівлі можна побачити у перерахуванні TenderStateMachineStatus:

```
public enum SmartStateMachineStatus
    {
        New=0, // Нова закупівля
        Planned=1, // Заплановані торги
        AcceptanceOfOffers = 2, // Подача пропозицій
        DataProcessing = 3, // Аналіз пропозицій
        Completed =4 // Завершено
    }
```

Таким чином реалізовано кінцевий автомат станів для закупівлі.

3. Дизайн Web API

За допомогою онлайн інструменту для проектування та документації API – SwaggerEditor [9] згенеровано документацію основних методів та моделей. Інструмент призначений для конструювання API та автоматичної генерації шаблонного коду для основних мов програмування: C#, JavaSript, Python, Swift.

Web API реалізовано на основі технологій ASP .NET Core MVC [10], контролери виконують роль маршрутизаторів та перенаправляють запити на відповідні мікросервіси [11]. У табл. 8 наведено перелік основних методів дії для закупівлі.

Таблиця 8. Перелік основних методів Web API

Шлях	Опис	Повертаєме значення
-GET- Tender/GetList	Отримати список закупівель	StatusCode 200 – OK + список TenderDTO StatusCode 204 – NoContent StatusCode 500 – помилка сервера
-GET- Tender/GetTenderById/{tenderId}	Отримати закупівлю за ідентифікатором	StatusCode 200 – OK + TenderDTO StatusCode 404 – NotFound StatusCode 500 – помилка сервера
-POST-Tender/Upsert	Додати нову закупівлю або оновити існуючу	StatusCode 201 – Created StatusCode 403 – Forbidden (немає доступу) StatusCode 500 – помилка сервера
-DEL-Tender/Delete/{tenderId}	Видалити закупівлю	StatusCode 200 – OK StatusCode 403 – Forbidden (немає доступу) StatusCode 500 – помилка сервера
-PATCH-TenderState/ChangeState/{tenderId/{stateId}	Змінити статус тендера на заданий	StatusCode 200 – Created StatusCode 404 – Not Found StatusCode 500 – помилка сервера
-PATCH-TenderState/MoveToNextStage	Перевести закупівлю на наступний етап	StatusCode 200 – OK StatusCode 404 – Not Found StatusCode 500 – помилка сервера

В ході розробки було прийняте рішення викликати валідацію у методах контролерів на сервері – у місці, де дані першими приходять від клієнта. Таким чином забезпечується коректність даних з самого початку їх обробки.

Оскільки перенавантажувати моделі даних логікою валідації вважається недоцільним, була розроблена власна система валідації на основі методів розширення для моделей та набору правил. Кожна модель у системі матиме метод IsValid, який буде викликати всі правила валідації для моделі, повертатиме булеве значення успішності валідації, а також масив помилок у форматі ключ-значення, де ключ – це назва поля моделі, а значення – опис помилки. У разі неуспішної валідації клієнту буде повертатися масив помилок, що буде оброблено та відповідним чином відображено у браузері.

Подібний підхід до процесу валідації дозволив розподілити відповідальність, очистити модель від непотрібних атрибутів або наслідування та покращити надійність програми.

4. Тестування

Тестування – це невд'ємна складова впровадження будь-якої програмної системи, тому під час розробки приділена значна увага покриттю коду тестами для швидкого виявлення та виправлення помилок, а також постійного контролю за роботоздатністю продукту. Під час тестування широко використовувались такі засоби, як unit- та mock-тести.

Для модульного тестування використовувалась бібліотека xUnit [12]. Це безкоштовний інструмент, що базується на відкритому до змін коді, який написано розробниками NUnit v2. Даний фреймворк призначено для тестування коду на C#, F#, VB.NET та інших мовах програмування .NET.

У фреймворку існує два поняття тестів: Fact та Theory. Fact – це окремий модульний тест, що не приймає параметрів. Theory – це тест, який приймає параметри та може містити декілька сценаріїв. Параметри до методу тесту можуть передаватися двома способами: через атрибути InlineData та MemberData.

За допомогою модульного тестування була перевірена робота системи валідації моделей, створені тестові методи для валідації кожної сутності у програмі. У кожному тесті на валідацію об'явлено два варіанта моделі: валідна та невалідна, а також два Theory-методи, які перевіряють валідацію. Також у даному тесті присутній Fact-метод, який перевіряє валідацію null-моделі.

За допомогою фреймворка Moq у системі розроблені тести для сервісів по роботі з закупівлями, лотами та позиціями. Для того, щоб перевірити CRUD-операції з основними сутностями системи створено мок-об'єкти відповідних сервісів-залежностей.

Moq – це фреймворк для створення моків для .NET, який розроблено з підтримкою специфічних можливостей .NET, таких як вирази LINQ та лямбди. Все це робить бібліотеку найбільш продуктивною, строго типізованою та дружньою до рефакторингу серед аналогів на ринку.

Наприклад, TenderController відповідає за основні операції для роботи із закупівлями, у нього є залежність – поле ITenderService, у якому знаходиться бізнес-логіка роботи з БД. Замість того, щоб при тестуванні створювати справжній об'єкт, ми створюємо мок ITenderService, який може працювати з тестовою базою даних або просто повертати потрібний результат. Після чого вже з використанням xUnit були написані модульні тести, які використовували створені у Moq моки сервісів.

Спираючись на результати проведених випробувань, можна зробити висновок про надійність системи та окремих її модулів. Відсоток покриття тестами становить 95 %, що підтверджує високу стійкість системи до збоїв.

Висновки

Реалізовано програмний засіб для автоматизації електронних закупівель з використанням новітніх технологій RESTful API та відкритих специфікацій

OpenAPI v3.0. Надано можливість авторизувати користувачів системи, а саме постачальників та замовників основуючись на стандарті Oauth. Бібліотека для роботи з розподіленим кешем Redis використовувалась для прискорення роботи системи та кешування даних на рівні доступу до БД. Для моніторингу стану закупівлі та переходу між станами реалізовано кінечний автомат станім за допомогою бібліотеки Stateless. Тестування розробленої системи проведено unit- та integration тестами. На відміну від комплексних систем автоматизації аукціонів по типу PHP Pro Bid, у яких клієнт є невідє'мною частиною сервера, що робить використання сервера неможливим для інших клієнтів (наприклад, мобільних додатків). У нашій системі електронних закупівель кожен модуль реалізовано як RESTFul API мікросервіс. Такий підхід до проектування серверної частини дозволив нам розподілити відповідальності між модулями та відділити сервер від клієнта. Використовуючи стандартний формат даних JSON, будь-який клієнт – веб-сайт або мобільний додаток, зможе повноцінно працювати з нашим API.

Література

1. Doroshenko A.Yu., Bodak B.V. (2019) .The impact and unforeseen challenges of E-procurement systems in Canada. Winter InfoCom Advanced Solutions. P. 9–10.
2. Business Dictionary, Reverse auction [Online] – Available from: https://financial-dictionary.thefreedictionary.com/Reverse+auction.
3. PHP Pro Bid Auction features [Online] – Available from: https://www.phpprobid.com/features/auctions.
4. iLance Auction Software [Online] – Available from: https://www.ilance.com/auction-software/.
5. Mubark A., Haggren A. (2017). Computer Science Optimization Of Reverse Auctions.
6. Stateless 3.0 – A State Machine library for .NET Core [Online] – Available from: https://www.hanselman.com/blog/stateless-30-a-state-machine-library-for-net-core/.
7. What is a RESTful API [Online] – Available from: https://searchapparchitecture.techtarget.com/definition/RESTful-API
8. Architectural Styles and the Design of Network-based Software Architectures [Online] – Available from: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
9. About Swagger specification [Online] – Available from: https://swagger.io/docs/specification/about/.
10. Repository and UnitOfWork in ASP.NET Core [Online] – Available from: http://www.c-sharpcorner.com/article/repository-pattern-in-asp-net-core/
11. Doroshenko A.Yu., Bodak B.V. (2020) The implementation of RESTful API for social media events platform. Summer InfoCom Advanced Solutions. P. 11–12.
12. Build, test, and deploy .NET Core apps [Online] – Available from: https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/dotnet-core?view=azure-devops.

References

1. Doroshenko A.Yu., Bodak B.V. (2019) .The impact and unforeseen challenges of E-procurement systems in Canada. Winter InfoCom Advanced Solutions. P. 9–10.
2. Business Dictionary, Reverse auction [Online] – Available from: https://financial-dictionary.thefreedictionary.com/Reverse+auction.
3. PHP Pro Bid Auction features [Online] – Available from: https://www.phpprobid.com/features/auctions.
4. iLance Auction Software [Online] – Available from: https://www.ilance.com/auction-software/.
5. Mubark A., Haggren A. (2017). Computer Science Optimization Of Reverse Auctions.
6. Stateless 3.0 – A State Machine library for .NET Core [Online] – Available from: https://www.hanselman.com/blog/stateless-30-a-state-machine-library-for-net-core/.
7. What is a RESTful API [Online] – Available from: https://searchapparchitecture.techtarget.com/definition/RESTful-API

8. Architectural Styles and the Design of Network-based Software Architectures [Online] – Available from: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
9. About Swagger specification [Online] – Available from: https://swagger.io/docs/specification/about/.
10. Repository and UnitOfWork in ASP.NET Core [Online] – Available from: http://www.c-sharpcorner.com/ article/repository-pattern-in-asp-net-core/
11. Doroshenko A.Yu., Bodak B.V. (2020) The implementation of RESTful API for social media events platform. Summer InfoCom Advanced Solutions. P. 11–12.
12. Build, test, and deploy .NET Core apps [Online] – Available from: https://docs.microsoft.com/en-us/azure/devops/pipelines/ecosystems/dotnet-core?view=azure-devops.

Одержано 20.01.2021

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень Інституту
програмних систем НАН України,
професор кафедри автоматики і
управління в технічних системах НТУУ
"КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в
українських виданнях – понад 190.
Кількість наукових публікацій в
іноземних виданнях – понад 80.
Індекс Гірша – 6.
http://orcid.org/0000-0002-8435-1451,

Бодак Богдан Вікторович,
аспірант кафедри автоматики і управління
в технічних системах НТУУ
"КПІ імені Ігоря Сікорського".
Кількість наукових публікацій в
українських виданнях – 2.

Місце роботи авторів:

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.
Тел.: (044) 526 3559.
E-mail: doroshenkoanatoliy2@gmail.com,
bohdan.bodak@outlook.com