ТЕХНІЧНІ НАУКИ

DOI: https://doi.org/10.32839/2304-5809/2020-12-88-20
УДК 004.415.2

Альохін А.В., Круглик В.С.
Херсонський державний університет

ПРОЄКТУВАННЯ ТА РОЗРОБКА МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ СИСТЕМИ БРОНЮВАННЯ

Анотація. В статті йде мова про використання системи бронювання з архітектурою мікросервісів, а саме система бронювання доріжок в басейні ХДУ, реалізована за допомогою веб-додатку. Дається уточнене визначення поняття мікросервісної архітекутри та її використання, призначення та принципи її реалізації. Сервіс системи бронювання басейну полягає в розробці та впровадженні програмного забезпечення для продажу і автоматичного аудиту наявності квитків та абонементів. Для розробки та реалізації системи було обрано платформу .NET Core. Розглянуто основні етапи проєктування та розроблення системи. Проаналізовано мікросервісну архітектуру та спроєктовано систему бронювання на основі мікросервісної архітектури. Розроблена інформаційна система виступає простим і ефективним засобом оформлення і бронювання послуги закладу відвідування басейну для студентів, викладачів університету та інших громадян.

Ключові слова: веб-система, мікросервіси, мікросервісна архітектура, .NET Core, API, система бронювання.

Alokhin Anton, Kruhlyk Vladyslav
Kherson State University

DESIGN AND DEVELOPMENT OF MICROSERVICE ARCHITECTURE FOR THE BOOKING SYSTEM

Summary. The article deals with the use of the booking system with the architecture of microservices, namely the system of booking tracks in the Kherson State University pool, implemented using a web application. The purpose of using the development system is to increase the efficiency of a separate structural unit "Pool" of Kherson State University. The automated booking system must solve a number of problems that usually arise during the booking process. The specified definition of the concept of microservice architecture and its use, purpose and principles of its realization are given. The microservice-based approach focuses mainly on business priorities and opportunities, while the monolithic approach is organized around the technological levels of user interfaces and databases. The microservice approach has become a trend in recent years as more and more businesses become flexible and move to DevOps. Microservices add unique consumer value by simplifying systems. The main stages of the booking system development process are considered. Examples of articles that testify to the relevance of research in the field of system development using microservice architecture are given. The service of the pool booking system is the development and implementation of software for the sale and automatic audit of tickets and season tickets. The .NET Core platform was chosen for the development and implementation of the system. The main stages of system design and development are considered. The microservice architecture is analyzed and the booking system based on the microservice architecture is designed. The developed information system is a simple and effective means of registration and booking of swimming pool services for students, university professors and other citizens. The project was implemented by means of the C# programming language, .NET Core platform, ASP.NET Core framework, Entity Framework Core framework, IdentityServer framework and Postgresql database. The structures of the Postgres databases for the microservices of the booking system and the description of the fields of its main tables were given. The scheme of interaction of microservices of the booking system and the scheme of data interconnection between databases of services are presented.

Keywords: web-system, microservices, microservice architecture, .NET Core, API, booking system.

Постановка проблеми. Проблема проектування та створення високоякісного програмного забезпечення надзвичайно важлива в сучасному інформаційному світі. З розвитком IT-галузі було знайдено багато різних підходів та концепцій до побудови складних програмних систем [6]. Показником добре структурованої програми є, звичайно, її архітектура, яка правильно описує предметну область і є формальною моделлю системи. Архітектурою можна вважати сукупність певних структурних компонентів, взаємопов'язаних, які визначають поведінку всієї системи. Головною метою архітектури є управління складністю, елегантний та відповідний спосіб відображення домену. Тривалий час провідне місце посідала так звана «монолітна архітектура». При такому підході вся система є монолітом, який фізично розташований на одній машині, працює в одному процесі та виконує всі ділові операції системи. Монолітне додаток масштабується лише за допомогою декількох окремих серверів з кожним окремим монолітом [5].

Але з часом були знайдені інші ідеї та підходи, саме такою стала архітектура мікросервісу. За допомогою мікросервісів ви можете розбити великі системи на менші компоненти, щоб відновити контроль над архітектурою. Ви можете внести невеликі зміни та додати окремі функціональні можливості за допомогою мікросервісів, розширивши один або набір компонентів та роз-

горнувши їх [1]. Завдяки архітектурі мікросервісу ваш додаток буде запускати різні служби як незалежні блоки, кожен зі своїм середовищем виконання, кодовою базою, потоками тощо.

Деякі з найбільш інноваційних та найвигідніших підприємств у світі – такі як Amazon, Netflix, Uber та Etsy – пояснюють величезний успіх своїх IT-ініціатив саме мікросервісами. З часом ці підприємства розібрали свої монолітні програми та переробили їх на архітектури на основі мікросервісів. Це допомогло швидко отримати величезні переваги, більшу спритність бізнесу та неймовірний прибуток.

Аналіз останніх досліджень і публікацій. Дослідження в області розробки додатків на мікросервісній архітектурі є актуальними й донині. У книзі І. Надареішвілі, Р. Мітра, М. Макларті та М. Амундсена [1] розглянуто проектно-орієнтований підхід до створення додатків на архітектури мікросервісів із вказівками щодо реалізації різних елементів та описано набір засобів та практик для вирішення практичних, організаційних та культурних проблем щодо прийняття мікросервісної архітектури. У статті Зеленін В.А. [2] досліджує основні підходи до побудови мікросервісної архітектури, а також існуючі концепцій до створення мікросервісних систем. У роботі Н. С. Жмуцької [4] розглянуто метод побудови веб-додатків на основі мікросервісної архітектури та визначено переваги та недоліки використання даного архітектурного стилю.

Виділення не вирішених раніше частин загальної проблеми. Використання системи бронювання дозволяє значною мірою покращити якість обслуговування клієнтів, оскільки, забезпечуючи їх бронювання в автоматичному режимі, скорочуються затрати часу на оформлення квитків чи абонементу, а також підвищується якість та ефективність роботи персоналу.

Сьогодні більшість людей віддають перевагу бронюванню послуг у мережі Інтернет, оскільки такий спосіб є більш зручним для клієнтів та має ряд переваг.

Розробка високонавантаженої та якісної системи бронювання полягає у необхідності створенні розподіленого серверного API архітектурою мікросервісів.

Мета статті. Мета дослідження полягає в створенні серверного API з архітектурою мікросервісів для системи бронювання басейну.

Викладення основного матеріалу дослідження. Реалізація проєкту по розробці серверного API для системи бронювання була здійснена за допомогою серверної мови програмування C#, програмної платформи .NET Core, об'єктно-реляційного фреймворку Entity Framework Core, фреймворку авторизації IdentityServer. В якості сховища інформації була обрана реляційна база даних PostgreSQL.

Процес розробки інформаційно-довідкової системи бронювання включає такі етапи:

1. Визначення теми, мети та цілей проєкту.

2. Аналіз та опис вимог до сервісу системи бронювання.

3. Проєктування системи та розподілу її на мікросервіси.

4. Проєктування структури баз даних мікросервісів системи.

5. Використання мови програмування C#, програмної платформи .NET Core, фреймворків Entity Framework Core та IdentityServer для розроблення системи бронювання.

Метою використання розроблювальної системи полягає у підвищення ефективності роботи окремого структурного підрозділу «Басейн» Херсонського державного університету. Автоматизована система бронювання повинна вирішувати ряд завдань, які зазвичай виникають у процесі резервування:

– уникнення «людського фактору», тобто найрозповсюдженіших помилок персоналу;

– перенавантаження чи, навпаки, простій ресурсів;

– зниження навантаження на співробітників компанії;

– полегшення процесу відслідковування роботи, формування звітів за певний проміжок часу,

– можливість надавати або позбавляти повноважень співробітників для роботи з системою, за необхідності.

Отже, для підтримки популярності сервісу, варто враховувати, що досвід взаємодії з ресурсом є надзвичайно важливим для успіху компанії.

Тобто, щоб клієнти із задоволенням користувалися послугами, які надаються сайтом компанії, зокрема, функцією бронювання як основною, необхідно, щоб сервіс був простий у використанні, а інтерфейс сайту – зручним та інтуїтивно зрозумілим.

Система бронювання повинна відображати реальну інформацію про:

– поточні тарифи;

– наявність вільних місць;

– можливість здійснення резервування.

Сервіс системи бронювання басейну полягає в розробці та впровадженні програмного забезпечення для продажу і автоматичного аудиту наявності квитків та абонементів.

Також система бронювання є гнучкою до вибори об'єктів доменої області. Тобто для часткового випадку системи бронювання басейну використовується доріжки басейну, але система може працювати з іншими об'єкта для гнучкості, наприклад бронювання квитків кінотеатру, тощо.

Для розробки системи бронювання було обрано підхід на основі мікросервісів, що орієнтований головним чином на бізнес-пріоритети і можливості, тоді як монолітний підхід організований навколо технологічних рівнів, призначених для користувача інтерфейсів і баз даних. Мікросервісний підхід став тенденцією в останні роки, так як все більше і більше підприємств стають гнучкими і переходять на DevOps [5]. Мікросервіси додають унікальну споживчу цінність шляхом спрощення систем. Розбиваючи вашу систему або додаток на безліч більш дрібних частин, ви реалізуєте спосіб зменшення дублювання, підвищення узгодженості та зменшення зв'язку між частинами, що робить ваші загальні частини системи більш зрозумілими, більш масштабованими і більш легкими для зміни [7].

В ході проєктування було вирішено що можна умовно розподілити систему на 2 основні частини: набір допоміжних сервісів, таких як точка для єдиного входу (API Gateway), а також незалежні самостійні мікросервіси. На наступному рисунку подано схему розподілу мікросервісів в системі бронювання (рис. 1).

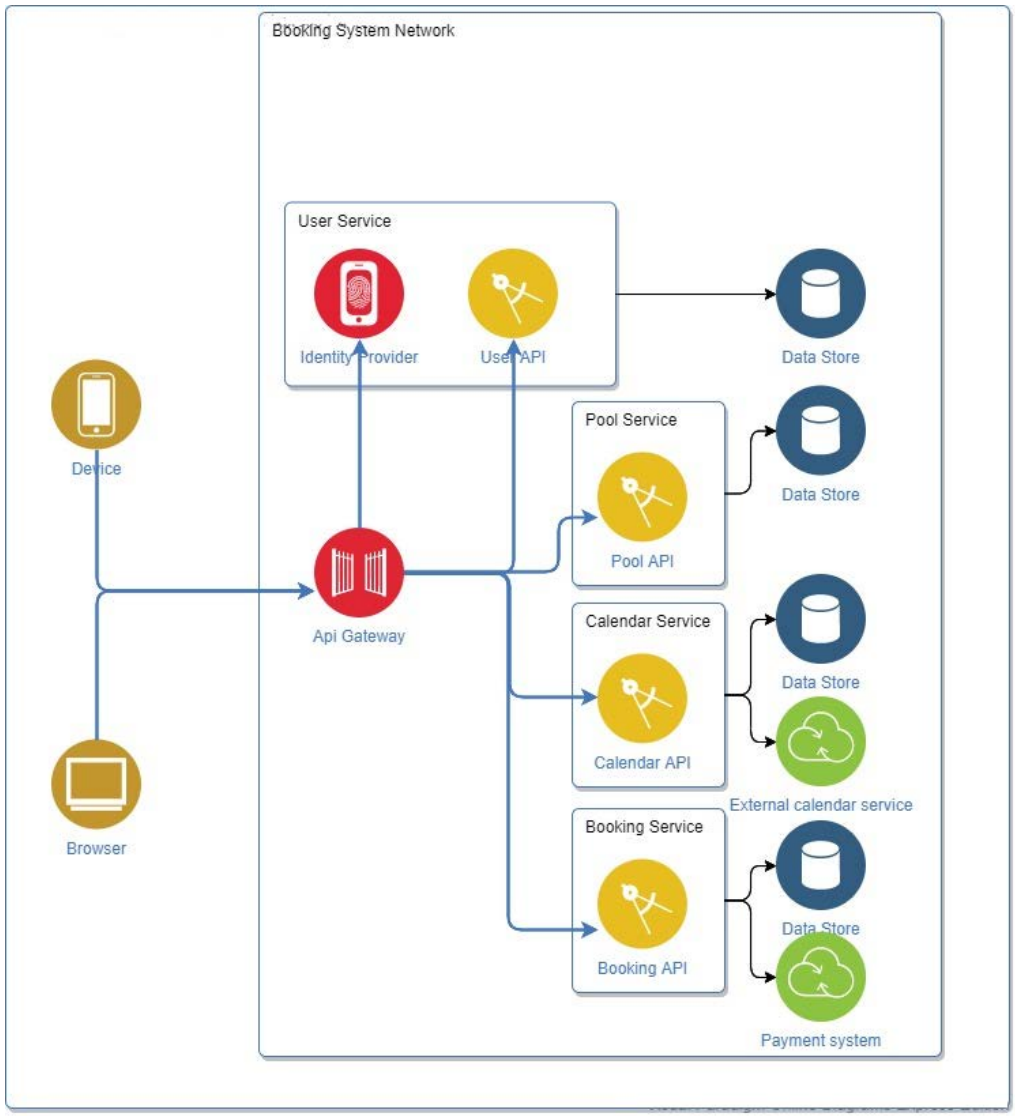

Рис. 1. Схема розподілу мікросервісів системи бронювання басейну

Всього в системі чотири мікросервіси: Pool Service, User Service, Calendar Service, Booking Service.

Pool Service є доменим сервісом системи бронювання, тобто цей сервіс відповідає за те у якій конкретній предметній області працює система бронювання. Сервіс надає можливість отримати дані про басейну, його контакти та місцезнаходження та інформацію беспосередньо про доріжки басейну, які є доменим об'єктом для системи бронювання.

User Service оперує даними користувачів та співпрацює із іншими сервісами в системі. Наприклад, при відображенні списку всіх замовлень користувача, ми йдемо до Booking Service. Також User Service представляє собою сервіс аутентифікації та авторизації, він імплементує Oauth2 та OpenId Connect протокол. Він розроблений, щоб надати загальний спосіб автентифікації запитів для всіх ваших програм, незалежно від того, чи є вони веб-сайтами, власними, мобільними або кінцевими точками API. Сервіс можна використовувати для реалізації єдиного входу (SSO) для декількох програм. Він може бути використаний для автентифікації фактичних користувачів за допомогою форм для входу та подібних користувацьких інтерфейсів, а також автентифікації на основі сервісу, який зазвичай включає видачу, перевірку та оновлення токенів без будь-якого інтерфейсу користувача.

Calendar Service оперує календарним графіком об'єкта (доріжки басейну). Зберігає в собі дані про розклад та календарні записи. Надає можливість запису календарної події різних видів (бронювання, технічне обслуговування, вихідний день, тощо) для об'єкта. Також сервіс використовується клієнтом для відображення вільних сеансів відвідування певних доріжок басейну.

Booking Service забезпечує систему функціоналом для бронювання об'єктів. Якщо можливо сервіс назначає резервацію певного об'єкту на певний час роблячи при цьому відповідний запис в Calendar Service. В майбутньому у цьому сервісі також плануються інтеграція з платіжною системою.

Також для успішної реалізації взаємодії мікросервісів та всієї внутрішньої роботи необхідно використовувати додаткові інструменти:

API Gateway. При розробці та створенні великих або складних програм на базі мікросервісів із кількома клієнтськими програмами хоро-

шим підходом може бути API Gateway. Це сервіс, який надає єдину точку доступу для певних груп мікросервісів. Це схоже на паттрн "Фасад" з об'єктно-орієнтованого дизайну, але в цьому випадку це частина розподіленої системи. Шаблон API Gateway також іноді називають "серверним інтерфейсом", оскільки ви створюєте його, думаючи про потреби клієнтської програми [3]. Тому API Gateway знаходиться між клієнтськими програмами та мікросервісами. Він діє як зворотний проксі-сервер, перенаправляючи запити від клієнтів до служб. Він також може надати додаткові наскрізні функції, такі як аутентифікація, припинення SSL та кеш-пам'ять.

Також система повинна бути побудована за шаблонами:

Агрегатор. Розбиваючи бізнес-функціонал на кілька менших логічних фрагментів коду, стає необхідним подумати про те, як співпрацювати дані, що повертаються кожним сервісом [3]. Цю відповідальність не можна покласти на споживача, оскільки тоді йому може знадобитися зрозуміти внутрішню реалізацію програми виробника. Паттерн агретор допомагає вирішити цю проблему. У ньому йдеться про те, як ми можемо узагальнити дані різних служб, а потім надіслати остаточну відповідь споживачеві. Для цього будемо використовувати API Gateway який розділить запит на декілька мікросервісів та буде агрегувати дані перед тим, як відправити їх споживачеві.

База даних на сервіс. Повинна бути розроблена одна база даних на мікросервіс; вона повинна бути приватною лише для цього сервісу. До нього повинен мати доступ лише API мікросервісу. До нього не можуть отримати доступ інші сервіси безпосередньо. Наприклад, для реляційних баз даних ми можемо використовувати приватні таблиці на сервіси, схему на послугу або сервер баз даних на послугу. Кожен мікросервіс повинен мати окремий ідентифікатор бази даних, щоб можна було надати окремий доступ для встановлення бар'єру та запобігання користуванню іншими таблицями сервісу.

База даних Pool Service зберігає домені дані системи бронювання – дані про басейн. Має дві головні таблиці Pool та PoolTrack (рис. 2). Pool зберігає в собі основну інформацію про басейн: назву, місцезнаходження та контактну інформацію. Також таблиця має відношення один до багатьох с таблицею PoolTrack. Тобто басейн має кілька доріжок кожна з яких має своє ім'я (Name) та унікальний номер (PoolTrackNumber).

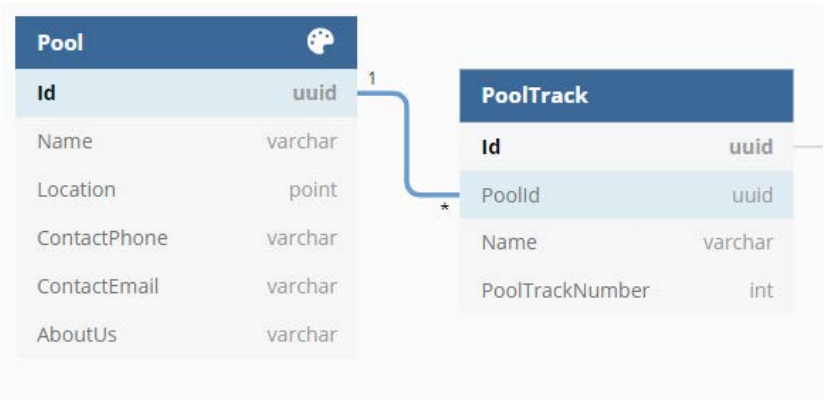

Рис. 2. Структура бази даних сервісу Pool Service

Для реалізації User Service використано систему ідентифікації ASP.NET Identity, зберігає всю інформацію про користувача у базі даних. ASP.NET Identity використовує Entity Framework Code First для реалізації всього свого механізму збереження. Тому проектування бази даних засновано

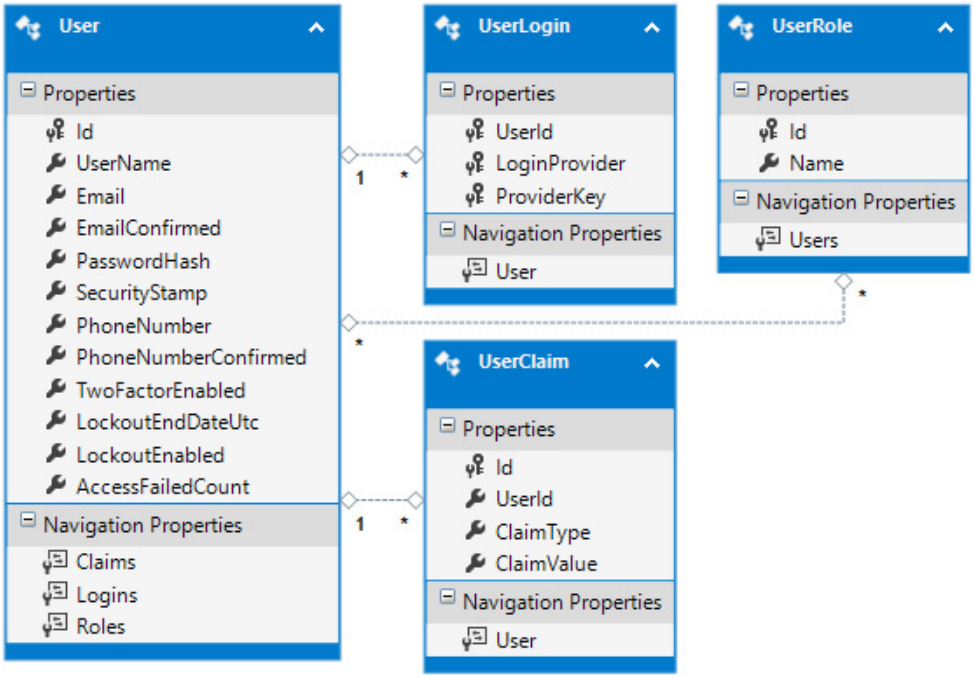

Рис. 3. Структура бази даних сервісу User Service

на структурі бази даних яка постачається використанням ASP. NET Identity. Містить в собі такі типові таблиці (рис. 3).

User – Зареєстровані користувачі вашого системи. Включає ідентифікатор користувача та ім'я користувача. Може містити хешований пароль, якщо користувачі входять з обліковими даними, які є специфічними для сайту (а не використовують облікові дані із зовнішнього сайту, такого як Facebook), та штамп безпеки (SecurityStamp), щоб вказати, чи щось змінилося в облікових даних користувача. Також може містити адресу електронної пошти, номер телефону, увімкнено двофакторну аутентифікацію, поточну кількість невдалих входів та чи заблоковано обліковий запис.

UserRole – Група юзерів з своїми правами. Включає ідентифікатор ролі та ім'я ролі (наприклад, "Адміністратор" або "Користувач").

UserClaim – Набір тверджень (або претензій) щодо користувача, які представляють ідентичність користувача. Може забезпечити більший вираз особистості користувача, ніж можна досягти за допомогою ролей.

UserLogin – Інформація про зовнішнього постачальника автентифікації (наприклад, Facebook), який слід використовувати під час входу в систему користувача.

База даних Calendar Service має такі основні таблиці (рис. 4).

Schedule – розклад для об'єкту (доріжки басейну), означає час старту та час кінця роботи об'єкту для певного дня. Розклад наперед створюється засобами Calendar Service. CalendarRecord – означає, що для певного об'єкта (доріжки басейну) була зроблена запис в календарі з якоїсь

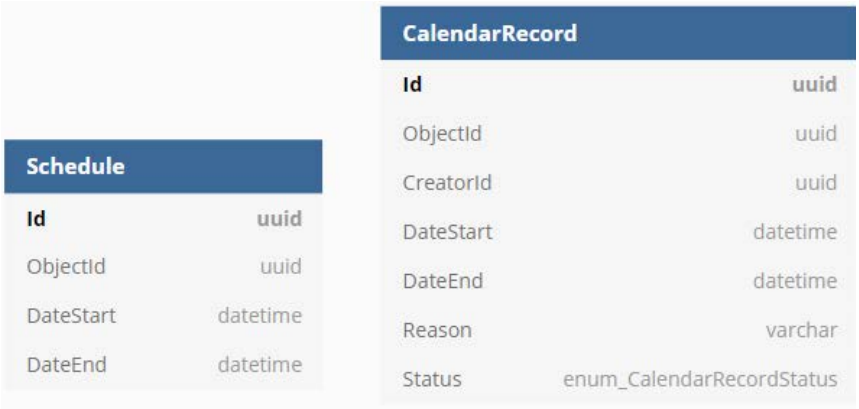

Рис. 4. Структура бази даних сервісу Calendar Service

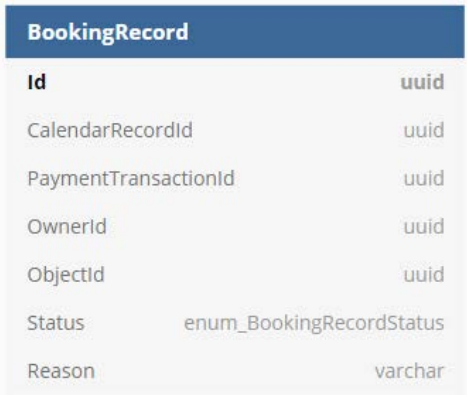

Рис. 5. Структура бази даних сервісу Booking Service

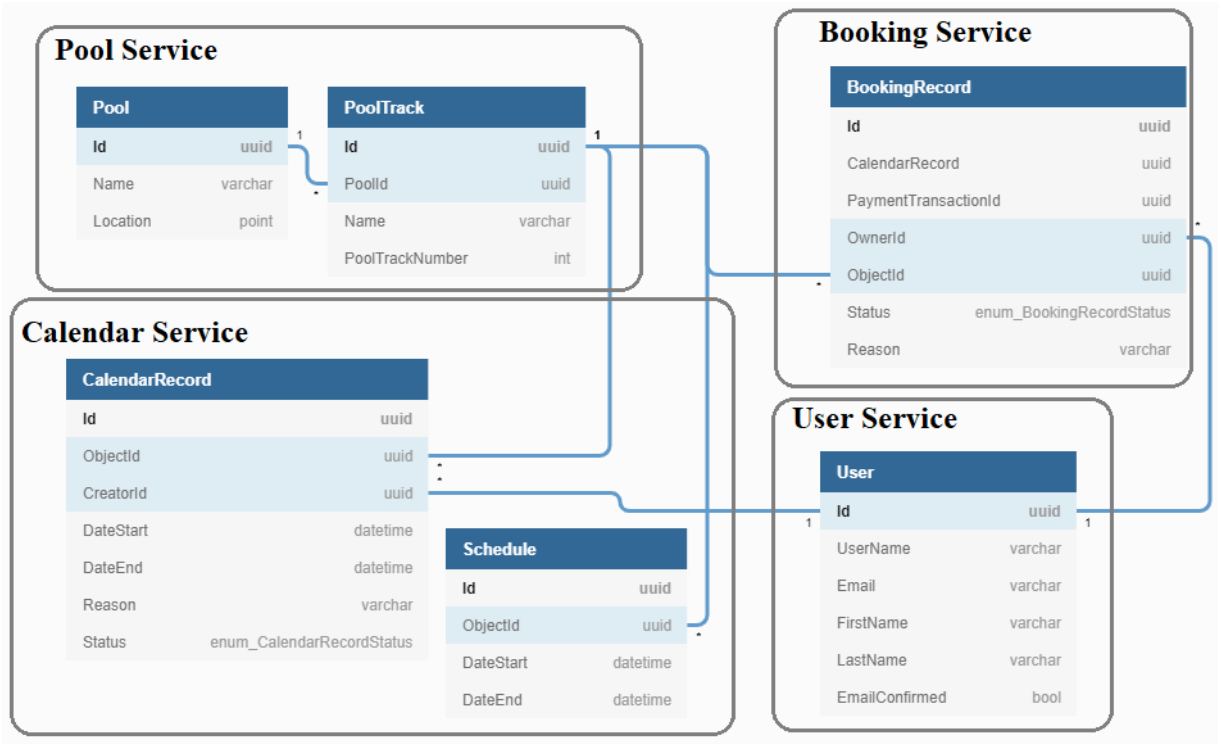

Рис. 6. Схема взаємозв'язку даних між базами даних сервісів

причини (має відповідний стовбець Reason) – може бути забронъований (Booked), знаходитися на технічному обслуговуванні (Maintenance), тощо. CreatorId позначає ким був зроблений запис, значення якого дорівнює ідентифікатору відповідного юзера в User Service.

Booking Service відповідно зберігає дані про зроблені бронювання в системі. Найголовніша таблиця в базі сервісу BookingRecord (рис. 5) пов'язує об'єкт бронювання (ObjectId), календарний запис (CalendarRecordId) та власника бронювання (OwnerId). Також закладена можливість у майбутньому запровадити платежі для бронювання (PaymentTransactionId).

Як було зазначено раніше кожен сервіс має свою окрему базу даних. Дані в сервісах слабо пов'язані між собою. Це означає, що не має зав'язків, які забезпечують цілісність, між таблицями різаних сервісів. Проте один сервіс может оперувати індетифікатор деякої сутності яка зберігається в іншому сервісі. Це ми можемо побачити на схемі взаємозв'язку даних між базами даних сервісів (рис. 6). Schedule, CalendarRecord та BookingRecord таблиці мають стовбець ObjectId, який означає домений об'єкт в системі бронювання, в нашому випаду – доріжку басейну та дорівнює PoolTrack.Id. Також CalendarRecord та BookingRecord посилаються на User таблицю через властивості CreatorId та OwnerId відповідно. А BookingRecord посилається на CalendarRecord відповідно через CalendarRecordId.

Висновки і пропозиції. Під час дослідження було створено серверний API для системи бронювання з використанням мікросервісної архітектури. Представлені результати дозволяють застосовувати розроблену систему для спрощення та автоматизування процесу бронювання доріжок для користувачів окремого структурного підрозділу «Басейн» Херсонського державного університету.

Система була розроблена з використанням архітектури мікросервісів що надало системі такі переваги як легке масштабування та гнучкість, незалежне розгортання, стійкість системи до відмов, гетерогенність технологій, модульність.

В перспективі розвитку системи її можна буде використовувати не тільки для предметної області бронювання доріжок басейну, а й для певної групи інших закладів, що надасть можливість використовувати систему як унікальне рішення для автоматизування процесу бронювання.

Список літератури:

1. Nadareishvili I. Microservice Architecture: Aligning Principles, Practices, and Culture / I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen. O'Reilly Media, 2016. 146 p.
2. Microservices. URL: http://martinfowler.com/articles/microservices.html (дата звернення: 02.11.2020).
3. Using an API Gateway. URL: https://nginx.com/blog/buildingmicroservices-using-an-api-gateway/ (дата звернення: 02.11.2020).
4. Жмуцька Н.С. Побудова веб-додатків на основі мікросервісної архітектури : Матеріали XLVIII науково-технічної конференції підрозділів ВНТУ. Вінниця, 13-15 березня 2019 р. 5 с.
5. Зеленін В.А. Основні принципи побудови мікросервісних систем. *Міжнародний науковий журнал "Інтернаука"*. 2017. № 11. 18 с.
6. Лен Басс. Архитектура программного обеспечения на практике (3-е изд.). Эддисон-Уэсли Профессионал, 2012. 102 с.
7. Ньюмен С. Создание микросервисов. Санкт-Петербург : Питер, 2016. 304 с.

References:

1. Nadareishvili I. (2016) Microservice Architecture: Aligning Principles, Practices, and Culture / I. Nadareishvili, R. Mitra, M. McLarty, M. Amundsen. O'Reilly Media, p. 146.
2. Microservices. Access mode: http://martinfowler.com/articles/microservices.html (accessed 02.11.2020).
3. Using an API Gateway. Access mode: https://nginx.com/blog/buildingmicroservices-using-an-api-gateway/ (accessed 02.11.2020).
4. Zhmutska N.S. (2019) Construction of web-applications based on microservice architecture: Proceedings of the XLVIII Scientific and Technical Conference of VNTU. Vinnytsia, March 13-15, p. 5.
5. Zelenin V.A. (2017) Basic principles of building microservice systems. *International Scientific Journal "Internauka",* no. 11, p. 18.
6. Len Bass (2012) Software architecture in practice (3rd ed.). Addison-Wesley Professional, p. 102.
7. Newman S. (2016) Creation of microservices. Sankt-Peterburg: Peter, p. 304.