https://doi.org/10.30857/2786-5371.2021.6.1

УДК 004.94; 681.3

[1]ПАНАСЮК О. І., [1]ПЛЕСКАЧ В. Л.,
[2]СТАЦЕНКО В. В., [3]ХОМАЗЮК В. А.
[1]Київський національний університет імені Тараса Шевченка, Україна
[2]Київський національний університет технологій та дизайну, Україна
[3]Національний медичний університет імені О.О. Богомольця, Київ, Україна

РОЗРОБКА МЕДИЧНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ МЕДИЧНИХ ЗАКЛАДІВ ПЕРВИННОЇ ЛАНКИ

Мета. *Розробка веб-застосунку з функціональним модулем проведення лікарських прийомів за стандартом ICPC2 для медичних закладів первинної ланки.*

Методика*. Для реалізації серверного застосунку з RESTful архітектурою було обрано мову C# 8.0 і фреймворк ASP.Net Core 5.0. У якості бази даних обрано СУБД MySql. Для розробки клієнтської частини використовувались HTML5 SASS, JavaScript, React та Redux.*

Результати*. Досліджені теоретичні основи і бізнес-процеси медичних інформаційних систем. Досліджені основні принципи побудови сучасної інформаційної системи. Спроектовано та реалізовано медичну інформаційну систему з високим показником надійності та швидкодії. Розроблений веб-застосунок з клієнт-серверною архітектурою*

Наукова новизна*. Виявлено особливості сучасних прикладних медичних інформаційних систем. Розглянуто можливості медичних інформаційних, як основного засобу зберігання медичних даних. Відпрацьовано процес ведення медичного прийому за стандартом ICPC2 для закладів первинної ланки. Досліджено теоретичні основи побудови програмної інформаційної системи для поліклініки та амбулаторії..*

Практична значимість. *Спроектовано та реалізовано медичну інформаційну систему з високим показником надійності та швидкодії, інтерфейсом зрозумілим для всіх вікових категорій користувачів. Розроблений веб-застосунок з клієнт-серверною архітектурою.*

Ключові слова: *веб-застосунок; клієнт-серверна архітектура; медична інформаційна система; система охорони здоров'я.*

Вступ. Розвиток сучасного суспільства відбувається під впливом глобалізації інформаційних процесів. Провідним проявом цих процесів є формування нового інформаційного простору, пов'язаного з появою мережі Інтернет. Формується нова культура в якій ключовими словами є інформація, інформатизація, Інтернет і віртуалізація. Поява нового інформаційного простору посприяла інтеграції інформаційних технологій у більшість соціальних, економічних і бізнес процесів. Загалом одним з основних проявів такої інтеграції є виникнення прикладних інформаційних систем, які покликані спростити життєдіяльність людей. В сучасній державі прикладні інформаційні системи створені майже для всіх сфер суспільного життя.

Охорона здоров’я є однією з найважливіших галузей діяльності держави. В сучасному інформаційному просторі охорона здоров’я є однією з найпоширеніших сфер впровадження провідних інформаційних технологій. В епоху розвинених інформаційних технологій державі необхідно забезпечити лікарів зручним і швидким засобом створення, зберігання і передачі медичної інформації. Для цього було створено медичні інформаційні системи.

Прикладні інформаційні системи з’явились зразу після виникнення персональних комп’ютерів, з того часу роль прикладних інформаційних систем в житті людей постійно збільшується. Медичні інформаційні систем існують вже давно, вони є невід’ємною частиною взаємодії між сучасним пацієнтом і лікарем. Також медичні інформаційні системи забезпечують зручні способи зберігання і обробки медичної інформації та створення звітів [1, 2].

В 2018 році в Україні розпочалась реформа системи охорони здоров'я, одним з основних законів реформи, був закон про обов'язкове подання електронних медичних даних до державної бази даних. На основі цих даних «Національна служба здоров'я України» регулює фінансування медичних закладів. Таким чином якщо раніше медичні інформаційні системи забезпечували лише прискорення і комфорт роботи, то тепер вони є обов'язковою частиною процесу отримання державного фінансування.

Аналіз попередніх досліджень. В 2018 році в Україні почав діяти законопроект про реформу системи охорони здоров'я в Україні. В рамках впровадження реформи було створено «Національну службу здоров'я України» (НСЗУ). Згідно з Законом, серед її основних функцій є забезпечення функціонування eHealth – системи охорони здоров'я [3].

Електронна інформаційно-телекомунікаційна система охорони здоров'я eHealth – забезпечує автоматизацію ведення обліку медичних послуг та управління медичною інформацією в електронному вигляді. У структуру eHealth входять центральна база даних та електронні медичні інформаційні системи, із автоматичним обміном даними через відкритий програмний інтерфейс (API). Після створення державного центрального компоненту eHealth з'явилось багато медичних інформаційних систем. На даний момент більше 20.

З найбільш популярних і досконалих слід зазначити: Helsi, Doctor Eleks, Emcimed, ПБЧ, MedStar, Medics

Система **Helsi** – медична інформаційна система, що є однією з найбільших в Україні та покриває велику долю ринку. В переліку партнерів компанії Helsi знаходяться лікарі, пацієнти, медичні заклади з усієї України, фармацевтичні компанії та страхові компанії [4].

За офіційними даними Національної служби здоров'я України через систему Helsi укладено 15,6 мільйона декларацій між лікарями та пацієнтами, що становить приблизно 44% всього обсягу укладених декларацій.

Система **Helsi** розроблена у форматі веб-застосунку. Веб застосунок складається з трьох частин: helsi.me; helsi.pro; reform.helsi.me. Helsi.me – це частина системи призначена для пацієнтів, helsi.pro – це частина системи призначена для лікарів і персоналу медичного заклад (доступне ведення прийому в форматі eHealth, з кодуванням діагнозів через систему ICPC2 або МКХ-11), reform.helsi.me – це частина системи призначена для адміністрації мед закладу.

Основними недоліками системи Helsi є:

- низька швидкість роботи підсистеми helsi.pro;
- не зрозумілий інтерфейс для лікарів які не мали досвіду роботи з медичними інформаційними системами;
- занадто громіздка для малих установ і ФОПів;
- інтерфейс у трьох підсистемах різний, тому перехід з reform.helsi.me на helsi.pro викликає у лікарів труднощі.

Doctor Eleks – це друга по розповсюдженості медична інформаційна система. Вона з'явилась ще у 2006 році і є однією з перших великих медичних інформаційних систем на території України [5]. Основними недоліками системи Doctor Eleks є:

- основний модуль системи для автоматизація робочого місця лікаря розповсюджується у форматі десктопного застосунку під операційну систему Windows, що робить неможливим його використання на інших платформах;
- десктопний застосунок має застарілий дизайн, незвичний і не зрозумілий для пересічного користувача;
- вартість впровадження системи є досить високою 400 доларів США на одне робоче місце на два роки без урахування витрат на устаткування, сервери та технічну підтримку.

Всі перераховані вище медичні інформаційні системи реалізовано у форматі веб-застосунку. З проведеного аналізу наявних медичних інформаційних систем, їх переваг та недоліків, а також враховуючи існуючі вимоги НСЗУ можна зробити наступні висновки:

- не мають модульної архітектури і тому не можуть бути налаштовані під вимоги конкретного закладу;
- через громіздкість не можуть швидко реагувати на нові вимоги НСЗУ і eHealth.

Постановка завдання. Основною задачею роботи є розробка веб-застосунку, що матиме функціональний модуль для проведення лікарських прийомів за стандартом ICPC2 для медичних закладів первинної ланки з високим показником надійності, швидкодії та інтерфейсом зрозумілим для всіх вікових категорій користувачів.

Результати дослідження. Проаналізувавши вимоги до медичної інформаційної системи, а також наявне у лікарів і пацієнтів ТЗ можна стверджувати що найбільш зручною платформою буде веб-застосунок. Архітектурою веб-застосунку обрано клієнт-серверну архітектуру (рис. 1).

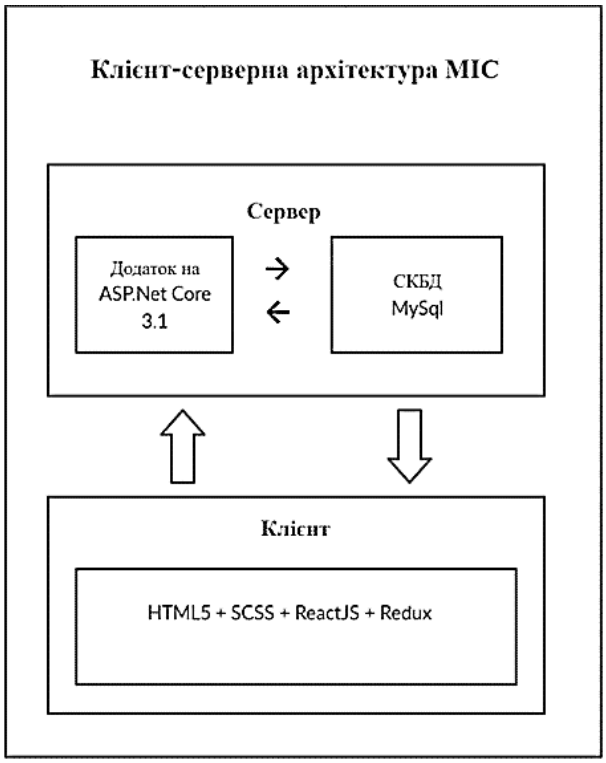

Рис. 1. **Схема роботи клієнт-серверної архітектури у медичній інформаційній системі**

Для створення клієнтської частини було обрано архітектуру односторінковий додаток (single-page application, SPA). Це веб-додаток або веб-сайт, який взаємодіє з веб-браузером, динамічно переписуючи поточну веб-сторінку новими даними з веб-сервера, замість стандартного способу, тобто завантаження цілих нових сторінок. Мета – швидші переходи, які дозволяють веб-сайту відчувати себе як рідний додаток. У SPA всі необхідні HTML, JavaScript та CSS-код отримують браузер із завантаженням однієї сторінки, або відповідні ресурси динамічно завантажуються та додаються на сторінку за необхідності, як правило, у відповідь на дії користувача. Сторінка не завантажується в будь-який момент процесу, а також не передає керування на іншу сторінку, хоча хеш місцеположення або API історії HTML5 можна використовувати для забезпечення сприйняття та навігації окремих логічних сторінок у програмі. SPA повністю завантажується при початковому завантаженні сторінки, а потім

частини сторінок замінюються або оновлюються новими фрагментами сторінок, завантаженими з сервера на вимогу. Щоб уникнути зайвого завантаження невикористаних функцій, SPA часто поступово завантажуватиме більше функцій, коли вони стануть необхідними, або невеликі фрагменти сторінки, або повно-екранні модулі [6].

Для реалізації серверного застосунку з RESTful архітектурою було обрано мову C# 8.0 і фреймворк ASP.Net Core 5.0. У якості бази даних обрано СУБД MySql. ASP.NET Core - вільно розповсюджений крос-платформенний фреймворк для створення веб-додатків з відкритим вихідним кодом. Дана платформа розроблена компанією Майкрософт і має більшу швидкодію порівняно з ASP.NET. Має модульну структуру та сумісний з такими операційними системами як Windows, Linux та macOS. Не зважаючи на те, що це новий фреймворк, побудований на новому веб-стеці, він має високу сумісність з концепціями з ASP.NET. Додатки ASP.NET Core підтримують паралельне управління версіями, в яких є різні програми, які працюють на одночасних комп'ютерах, і можуть орієнтуватися на різні версії ASP.NET Core. Це було неможливо в попередній версії ASP.NET [7, 8].

У якості СУБД було обрано MySQL, його було порівняно з MongoDB. MySQL – вільна система управління базами даних, яка є власністю компанії Sun Mycrosystem. СУБД розповсюджується по ліцензії GNU General Public License та під власною комерційною ліцензією. MySQL підтримує багато типів таблиць, вона є мультиплатформенною СУБД. Завдяки широким можливостям, а також великій швидкодії MySQL часто використовується в якості СУБД, що забезпечує роботу динамічного Web-сайту в мережі Інтернет [9–11].

Порівняння реляційних і не реляційних баз даних [9–11].

Мова. Реляційні бази даних використовують структурований мову запитів (Structured Query Language, SQL) для визначення і обробки даних. З одного боку, це відкриває великі можливості для розробки: SQL один з найбільш гнучких і поширених мов запитів, так що його вибір дозволяє мінімізувати ряд ризиків, і буде особливо до речі, якщо має бути робота з комплексними запитами. З іншого боку, в SQL є ряд обмежень. Побудова запитів на цій мові зобов'язує визначати структуру даних і, як у випадку з Містом А, подальша зміна структури даних може бути згубним для всієї системи. Нереляційні бази даних, в свою чергу, пропонують динамічну структуру даних, які можуть зберігатися кількома способами: орієнтовано по колонках, документ-орієнтовано, в вигляді графів або на основі пар «ключ-значення». Така гнучкість означає наступне:

- можна створювати документи, не ставлячи їх структуру заздалегідь;
- кожен документ може мати власну структурою;
- у кожної бази даних може бути власний синтаксис;
- можна додавати поля прямо під час роботи з даними.

Масштабованість. У більшості випадків SQL бази даних вертикально масштабовані, тобто ви можете збільшувати навантаження на окремо взятий сервер, нарощуючи потужність центральних процесорів, обсяги ОЗУ або системи зберігання даних. А NoSQL бази даних горизонтально масштабованих. Це означає, що ви можете збільшувати трафік, розподіляючи його або додаючи більше серверів до вашої СУБД. Все одно, що додавати більше поверхів до вашого будинку, або додавати більше будівель на вулицю. У другому випадку, система може стати куди більші і потужніші, роблячи вибір NoSQL бази даних віддається перевага для великих або постійно мінливих структур даних.

Структура. У реляційних СУБД дані представлені у вигляді таблиць, в той час як в нереляційних – у вигляді документів, пар «ключ-значення», графів або wide-column сховищ.

Відомо, що не реляційні бази даних, на прикладі MongoDB, значно повільніші при оновленні даних з індексом навіть на невеликих кількостях та читанні даних.

Отже, виходячи з результатів наведених вище порівнянь було обрано СУБД MySQL через швидкодію, велику розповсюдженість, широку підтримку і високу сумісність з об'єктно-орієнтованими мовами програмування.

Для вирішення завдань проектованої системи було проаналізовано декілька якісних наявних додатків з операціями та можливостями ведення сімейного бюджету та виділено функціонал, який найкраще підходить для проектованої системи, не ускладнює роботу користувачу та не зменшує продуктивність всього веб-сайту (табл. 1).

Таблиця 1

Функціонал веб-додатку

Назва функціоналу	Опис функціоналу
Авторизація користувача	Для можливості ідентифікації користувачів та збереженням їх даних на сервері кожен користувач повинен пройти авторизації та підтвердити свої реєстраційні дані.
Реєстрація	Функція яка дозволяє створювати облікові записи для кожного користувача з розділенням по ролям.
Створення медичного закладу	Функція створення медичного закладу дозволяє керівнику зареєструвати свій заклад.
Додавання лікарів до медичного закладу	Функція додавання лікарів до медичного закладу дозволяє керівнику зареєструвати свій заклад.
Редагування медичного закладу	Функція редагування гаманця дозволяє проводити зміни в назві адресі та інших властивостях медичного закладу.
Пошук лікарів	Функція пошуку лікарів дозволяє пацієнтам знаходити лікарів за прізвищем, а також фільтрувати результати.
Назва функціоналу	Опис функціоналу
Запис на прийом до лікаря	Пацієнт може записатись на прийом до лікаря
Скасування прийому	Пацієнт або лікар можуть скасувати запис на прийом
Ведення прийому	Лікар може провести прийом з пацієнтом який був записаний до нього. В рамках ведення прийому лікар може додавати причини, діагнози, послуги з кодуванням ICPC2

Схема бази даних (рис. 2):

- Користувачі – таблиця для зберігання облікових записів користувачів, загальних даних про них (ПІБ, стать, дата народження, телефон).
- Організації – таблиця для зберігання інформації про організації, їх адресу і код ЄДРПОУ.
- Прийоми – таблиця для зберігання інформації про прийоми зі зв'язком з таблицею користувачів.
- Довідник ICPC2 – таблиця для зберігання довідника ICPC2 кодів.

Вся інформація складається з набору однотипних записів розташованих один за одним. Дані в цих таблицях можливо додавати, видаляти або змінювати. Також в кожній таблиці є набір іменованих полів, які зберігають корисну інформацію починаючи від імені користувача та закінчуючи довідником ICPC2 кодів.

В серверній частині веб-застосунку для реалізації механізму генерації і отримання доступних часових проміжків для запису на прийом до лікаря було застосовано один з популярних патернів проектування – фасад [12]. За допомогою патерну фасад можна істотно спростити доступ до системи класів або методів. Це допомагає розділити шари логіки застосунку. У випадку підсистеми генерації проміжків часу для запису фасад дозволяє

приховати всю реалізацію від шару взаємодії (API) тим самим розділяючи рівні API і бізнес-логіки, що в подальшому спростить підтримку, розширення і рефакторинг системи (рис. 3).

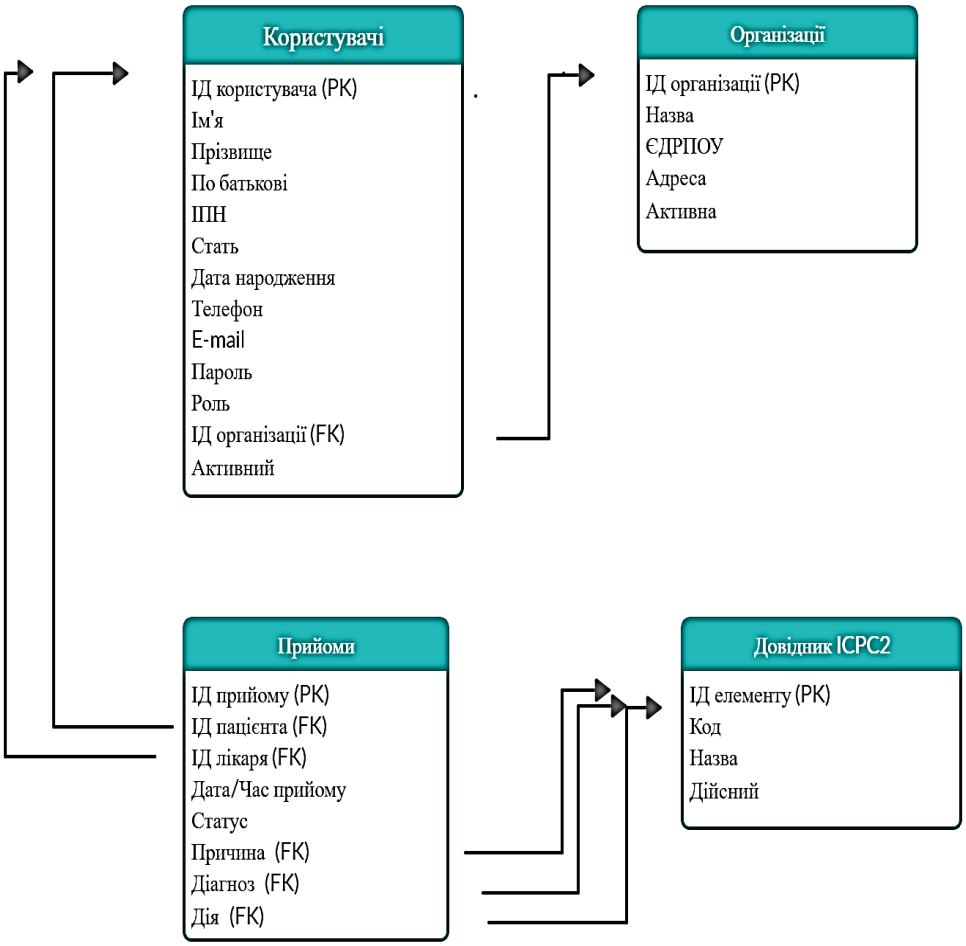

Рис. 2. **Схема бази даних**

```
namespace HealthyCountry.Services
{
    3 usages  1 inheritor  Oleksandr Panasiuk  1 exposing API
    public interface IAppointmentFacade
    {
        17 usages  1 implementation  Oleksandr Panasiuk
        Task<List<Appointment>> GetAppointmentsAsync(string doctorId, DateTime dateFrom);
    }
}
```

Рис. 3. **Інтерфейс фасаду отримання проміжків часу для запису**

Рівень API отримує доступ до підсистеми генерації і отримання доступних часових проміжків для запису на прийом до лікаря через метод GetAppointmentsAsync інтерфейсу IAppointmentFacade (рис. 4).

Клас AppointmentService реалізує інтерфейс фасаду, надаючи рівню API лише одну просту точку входу до підсистеми бізнес-логіки.

```
public class AppointmentService : IAppointmentFacade
{
    private readonly ApplicationDbContext _dbContext;

    1 usage  Oleksandr Panasiuk
    public AppointmentService(ApplicationDbContext dbContext)
    {
        _dbContext = dbContext;
    }

    0+17 usages  Oleksandr Panasiuk
    public Task<List<Appointment>> GetAppointmentsAsync(string doctorId, DateTime dateFrom)
    {
        var numberOfSlotsToCreate :int = GetNumberOfSlotsToCreate(doctorId, dateFrom);
        if (numberOfSlotsToCreate != 0)
        {
            CreateAppointmentSlots(doctorId, dateFrom, numberOfSlotsToCreate);
        }

        CreateSlotsForCancelledAppointments(doctorId, dateFrom);
        return _dbContext.Appointments.AsNoTracking()
            .Where(x :Appointment => x.EmployeeId == doctorId && x.DateTime.Date >= dateFrom.Date && x.Status!=AppointmentStatuses.CANCELED)
            .OrderBy(x :Appointment => x.DateTime).ToListAsync(); // Task<List<...>>
    }
```

Рис. 4. **Клас і метод що реалізують інтерфейс фасаду**

У клієнтській частині веб-застосунку для спрощення роботи з формами вводу було розроблено універсальний компонент вводу. Основним функціоналом компоненту є створення відповідного HTML елементу форми в залежності від вхідних параметрів.

За допомогою властивостей які передаються через props компонент можна конфігурувати як будь-яке з HTML полів для вводу, а також були додані окремі опції для асинхронного селекту, тобто випадаючого списку де опції завантажуються динамічно з серверу коли користувач вводить символи.

Для веб-застосунку розробленого для пацієнтів та лікарів було обрано мінімалістичний дизайн для зручності користування. Для користування системою користувачу необхідно увійти у свій обліковий запис. Система дозволяє директору зареєструвати свій мед. заклад і додати співробітників. Для пацієнтів створено окремий зручний інтерфейс реєстрації.

Для виконання тестування веб-сайту було використано браузер Google Chrome з інструментами розробника а саме, "Network" та "Lighthouse".

За результатами тестування в інструменті "Lighthouse" було отримано наступні значення: Perfomance (продуктивність) − 93/100; Accessibility (доступність) − 97/100; Best Practices (використання найкращих рішень) − 93/100; SEO (оптимізація під пошукові системи) – 100/100.

Ці дані говорять, що веб-сайт відповідає всім нормам та правилам створення веб додатків та буде частіше пропонуватись в пошукових системах.

Висновки. В результаті було розроблено медичну інформаційну систему. В процесі виконання роботи було досліджено теоретичні основи і бізнес-процеси медичних інформаційних систем. Досліджені всі основні принципи, необхідні для побудови сучасної інформаційної системи, порівнянні два види баз даних і обрано оптимальний для виконання поставленого завдання. Переглянуті існуючі рішення в даній сфері і сформульовано технічні вимоги. Спроектовано та реалізовано медичну інформаційну систему з високим показником надійності, інтерфейсом зрозумілим для всіх вікових категорій користувачів. Також при

розробці системи було зроблено упор на швидкодію. Розроблений веб-застосунок з клієнт-серверною архітектурою.

References

1. Medychni informatsiini systemy – vprovadzhuiemo u vashomu medychnomu zakladi [Medical information systems – we implement in your medical institution]. URL: https://www.medsprava.com.ua/article/544-qqq-17-m4-10-04-2017--medichn-nformatsyn-sistemi-vprovadjumo-u-vashomu-medichnomu-zaklad.
2. Zahorodnii, H. M., Zynov'iev, Ye. S., Martynov, V. M., Shadura, V. M. (2005). HRID – nova obchysliuvalna tekhnolohiia dlia nauky [GRID – a new computing technology for science]. *Visnyk NAN Ukrainy = Bulletin of the NAS of Ukraine*, № 6, P. 17–19 [in Ukrainian].
3. eHealth: website. URL: https://ehealth.gov.ua/.
4. Helsi.me. Helsi: website. URL: https://helsi.me/.
5. Doctor Eleks: website. URL: https://ehealth.eleks.com/.
6. Albahari, J., Albahari, B. (2008). LINQ Pocket Reference: Learn and Implement LINQ for .NET Applications: O'Reilly Media. 174 p.
7. React 2020. React: website. URL: https://uk.reactjs.org/.
8. MongoDB 2020. MongoDB: website. URL: https://www.mongodb.com.
9. SQL protyv NoSQL na prymere MySQL y MongoDB [SQL versus NoSQL using MySQL and MongoDB as an example]. TProger: website. URL: https://tproger.ru/translations/sql-vs-nosql.
10. Chambers, J., Paquette, D., Timms, S. (2017). ASP.NET Core Application Development: Bulding an Application in Four Sprints (Developer Reference): Microsoft Press. P. 25–277.
11. Mark, J. (2019). Price C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Packt Publishing. 818 p.
12. Paterny proektuvannia – fasad [Design patterns – facade]. RefactoringGuru: website. URL: https://refactoring.guru/uk/design-patterns/facade [in Ukrainian].

Література

1. Медичні інформаційні системи – впроваджуємо у вашому медичному закладі. URL: https://www.medsprava.com.ua/article/544-qqq-17-m4-10-04-2017--medichn-nformatsyn-sistemi-vprovadjumo-u-vashomu-medichnomu-zaklad.
2. Загородній Г. М., Зинов'єв Є. С., Мартинов В. М., Шадура В. М. ГРІД – нова обчислювальна технологія для науки. *Вісник НАН України*. 2005. № 6. С. 17–19.
3. eHealth: веб-сайт. URL: https://ehealth.gov.ua/.
4. Helsi.me. Helsi: веб-сайт. URL: https://helsi.me/.
5. Doctor Eleks: веб-сайт. URL: https://ehealth.eleks.com/.
6. Albahari J., Albahari B. LINQ Pocket Reference: Learn and Implement LINQ for .NET Applications: O'Reilly Media, 2008. 174 p.
7. React 2020. React: веб-сайт. URL: https://uk.reactjs.org/.
8. MongoDB 2020. MongoDB: веб-сайт. URL: https://www.mongodb.com.
9. SQL против NoSQL на примере MySQL и MongoDB. TProger: веб-сайт. URL: https://tproger.ru/translations/sql-vs-nosql.
10. Chambers J., Paquette D., Timms S. ASP.NET Core Application Development: Bulding an Application in Four Sprints (Developer Reference): Microsoft Press, 2017. P. 25–277.
11. Mark J. Price C# 8.0 and .NET Core 3.0 – Modern Cross-Platform Development: Packt Publishing, 2019. 818 p.
12. Патерни проектування – фасад. RefactoringGuru: веб-сайт. URL: https://refactoring.guru/uk/design-patterns/facade.

PANASIUK OLEKSANDR
Department of Applied Information Systems
Taras Shevchenko National University of Kyiv, Ukraine
Scopus Author ID: 57289566900
E-mail: vohigi@gmail.com

PLESKACH VALENTYNA
Doctor of Economic Sciences, Professor
Department of Applied Information Systems
Taras Shevchenko National University of Kyiv, Ukraine
Scopus Author ID: 12039392900
E-mail: v_pleskach@ukr.net

STATSENKO VOLODYMYR
Doctor of Technical Sciences, Associate Professor
Department of Computer Engineering
and Electromechanics
Kyiv National University of Technologies
and Design, Ukraine
https://orcid.org/0000-0002-3932-792X
Scopus Author ID: 57210344190
Researcher ID: C-3646-2017
E-mail: statsenko.v@knutd.edu.ua

KHOMAZIUK VICTORIA
Candidate of Medical Sciences, Associate Professor
Department of Propaedeutics of Internal Medicine N2
Bogomolets National Medical University, Ukraine
https://orcid.org/0000-0003-3466-2574
Scopus Author ID: 6506224779
Researcher ID:
E-mail: khomaziuk.victoria@nmu.ua

[1]ПАНАСЮК А. И., [1]ПЛЕСКАЧ В. Л., [2]СТАЦЕНКО В. В., [3]ХОМАЗЮК В. А.
[1]Киевский национальный университет имени Тараса Шевченко, Украина
[2]Киевский национальный университет технологий и дизайна, Украина
[3]Национальный медицинский университет имени А.А. Богомольца, Киев, Украина

РАЗРАБОТКА МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ МЕДИЦИНСКИХ ЗАВЕДЕНИЙ ПЕРВИЧНОГО ЗВЕНА

Цель. *Разработка веб-приложения с функциональным модулем проведения врачебных приемов по стандарту ICPC2 для медицинских учреждений первичного звена.*

Методика. *Для реализации серверного приложения с RESTful архитектурой был выбран язык C#8.0 и фреймворк ASP.Net Core 5.0. В качестве базы данных выбрана СУБД MySql. Для разработки клиентской части использовались HTML5 SASS, JavaScript, React и Redux.*

Результаты. *Исследованы теоретические основы и процессы медицинских информационных систем. Исследованы главные принципы построения современной информационной системы. Спроектирована и реализована медицинская информационная система с высоким показателем надежности и быстродействия. Разработано веб-приложение с клиент-серверной архитектурой*

Научная новизна. *Выявлены особенности современных прикладных медицинских информационных систем. Рассмотрены возможности медицинских информационных как основного средства хранения медицинских данных. Отработан процесс ведения медицинского приема по стандарту ICPC2 для учреждений первичного звена. Исследованы теоретические основы построения программной информационной системы для поликлиники и амбулатории.*

Практическая значимость. *Спроектирована и реализована медицинская информационная система с высоким показателем надежности и быстродействия, интерфейсом понятным для всех возрастных категорий пользователей. Разработано веб-приложение с клиент-серверной архитектурой.*

Ключевые слова: *веб-приложение; клиент-серверная архитектура; медицинская информационная система; система здравоохранения.*

[1]PANASIUK O. I., [1]PLESKACH V. L., [2]STATSENKO V. V., [3]KHOMAZIUK V. A.
[1]Taras Shevchenko National University of Kyiv, Ukraine
[2]Kyiv National University of Technologies and Design, Ukraine
[3]Bogomolets National Medical University, Kyiv, Ukraine

DEVELOPMENT OF MEDICAL INFORMATION SYSTEM FOR PRIMARY MEDICAL INSTITUTIONS

Purpose. *Development of a web application with a functional module for conducting prescriptions according to the ICPC2 standard for primary care facilities.*

Methodology. *The C # 8.0 language and the ASP.Net Core 5.0 framework were chosen to implement a server application with the RESTful architecture. The MySql database is selected as the database. HTML5 SASS, JavaScript, React and Redux were used to develop the client part.*

Findings. *Theoretical bases and business processes of medical information systems are investigated. The basic principles of building a modern information system are studied. A medical information system with a high rate of reliability and speed has been designed and implemented. Developed a web application with a client-server architecture*

Originality. *Features of modern applied medical information systems are revealed. Possibilities of medical information as the main means of medical data storage are considered. The process of conducting medical reception according to the ICPC2 standard for primary care facilities has been worked out. Theoretical bases of construction of software information system for polyclinic and outpatient clinic are investigated.*

Practical value. *A medical information system with a high level of reliability and speed, an interface understandable for all age groups of users has been designed and implemented. Developed a web application with client-server architecture.*

Keywords*: web application; client-server architecture; medical information system; health care system.*