https://orcid.org/0000-0003-4326-3030

УДК 004.4

Д.В. РАТОВ*

МОДЕЛЬ МОДУЛЯ ІНТЕРФЕЙСУ КОРИСТУВАЧА ІНФОРМАЦІЙНОЇ WEB-СИСТЕМИ

*Східноукраїнський національний університет імені Володимира Даля, м. Сєвєродонецьк, Україна

Анотація. *У статті розглядаються архітектурний шаблон і методи побудови модуля для організації роботи віконної інформаційної web-системи формування медичних сертифікатів. Для вирішення даного завдання застосовані сучасні засоби веб-програмування і розробки програмного забезпечення: створений модуль із реалізацією об'єктів JavaScript – диспетчер вікон і клас віконного інтерактивного інтерфейсу, які дозволяють функціонувати web-додатку при організації хмарних технологій. Розглянуто програмну реалізацію і наведені результати практичного використання розробленого модуля з об'єктами і класами. Запропонований у роботі шаблон архітектури модуля для реалізації диспетчера та функціонала користувацького віконного інтерфейсу дозволив створити модуль інформаційної системи для організації бізнес-логіки при роботі з медичними сертифікатами. Модульний підхід позитивно позначився на швидкості призначеного для користувача інтерфейсу і можливості масштабування функціонала самої системи при реалізації хмарних технологій. Завдяки модульному підходу, час рендеринга web-форм із вихідними даними у більшості випадків є дуже незначним у порівнянні з часом, необхідним браузеру для розбору і відображення всього масиву даних бази, що позитивно позначається на швидкості призначеного для користувача інтерфейсу. Величезне практичне значення має генерація web-форм, тому що налагодження помилок у цій програмній структурі є болючим місцем багатьох фреймворків. Результати використання показали, що механізм модульного створення диспетчера і віконного інтерактивного інтерфейсу web-форм не тільки органічно вписується в уже існуючі технології побудови web-додатків, але і сам має достатній потенціал, щоб у майбутньому стати ядром хмарних технологій розробки багатокористувацьких інформаційних систем і web-сервісів.*

Ключові слова: *модель, модуль, клас інтерфейсу користувача, диспетчер вікон, хмарні обчислення, фреймворк.*

Аннотация. *В статье рассматриваются архитектурный шаблон и методы построения модуля для организации работы оконной информационной web-системы формирования медицинских сертификатов. Для решения данной задачи применены современные средства веб-программирования и разработки программного обеспечения: создан модуль с реализацией объектов JavaScript – диспетчер окон и класс оконного интерактивного интерфейса, которые позволяют работать web-приложения при организации облачных технологий. Рассмотрена программная реализация и приведены результаты практического использования разработанного модуля с объектами и классами. Предложенный в работе шаблон архитектуры модуля для реализации диспетчера и функционала пользовательского оконного интерфейса позволил создать модуль информационной системы для организации бизнес-логики при работе с медицинскими сертификатами. Модульный подход положительно сказался на скорости пользовательского интерфейса и возможности масштабирования функционала самой системы при реализации облачных технологий. Благодаря модульному подходу, время рендеринга web-форм с исходными данными в большинстве случаев очень незначительно по сравнению со временем, необходимым браузеру для разбора и отображения всего массива данных базы, что положительно сказывается на скорости пользовательского интерфейса. Огромное практическое значение имеет генерация web-форм, так как налаживание ошибок в этой программной структуре является больным местом многих фреймворков. Результаты использования показали, что механизм модульного создания диспетчера и оконного интерактивного интерфейса web-форм не только органично вписывается в уже существующие технологии построения web-приложений, но и сам имеет достаточный потенциал, чтобы в будущем стать ядром облачных технологий разработки многопользовательских информационных систем и web- сервисов.*

Ключевые слова: *модель, модуль, класс интерфейса пользователя, диспетчер окон, облачные вычисления, фреймворк.*

Abstract. *The paper considers the architectural template and methods of building a module for organizing the work of the window information web-system for the formation of medical certificates. To solve this problem, modern tools of web programming and software development are used: a module with the implementation of JavaScript objects – window manager and a class of window interactive interface, which allow the web application to function in the organization of cloud technologies. The software implementation is considered and the results of practical use of the developed module with objects and classes are given. The proposed architecture template of the module for the implementation of the manager and the functionality of the user window interface allowed to create a module of the information system for the organization of business logic when working with medical certificates. The modular approach has had a positive effect on the sensitivity of the user interface and the ability to scale the functionality of the system itself in the implementation of cloud technologies. Due to the modular approach, the rendering time of web-forms with the original data in most cases is very small compared to the time required by the browser to parse and display the entire database array, which has a positive effect on the sensitivity of the user interface. The generation of web-forms is of great practical importance, because debugging in this software structure is a weak point for many frameworks. The results of use showed that the mechanism of modular creation of the manager and the window interactive interface of web-forms not only organically fits into already existing technologies of construction of web-applications, but also has sufficient potential to become the core of cloud technologies of development of multiuser information systems and web- services.*
Keywords: *pattern, module, user interface class, window manager, cloud computing, framework.*

DOI: 10.34121/1028-9763-2020-4-74-81

1. Вступ

Безперервний розвиток інформаційних технологій і технічних характеристик мережі Інтернет сприяв підвищенню інтересу до розробки інформаційних систем із використанням хмарних технологій [1]. Хмарні обчислення (від англ. Cloud computing) спрямовані на надання користувачу комп’ютерних ресурсів і потужностей у вигляді інтернет-сервісів і інформаційних web-систем.

На сьогоднішній день існують web-технології, і на їх базі розроблені бібліотеки і фреймворки для створення web-додатків та інтерфейсів користувача, призначених для роботи інформаційних систем у браузерах. Процеси стандартизації мов HTML [2], CSS [3] і javascript [4] дозволили досягти не тільки високого ступеня кросплатформних технологій, призначених для користувача інтерфейсів, але і досить високого ступеня кросбраузерності, тому використання відповідних стандартів при побудові web-додатків стало домінуючим підходом.

При розробці модульних інформаційних систем одних тільки стандартів виявляється недостатньо. Потрібні шаблони проєктування, бібліотеки стандартних елементів управління, засоби підтримки логіки роботи систем (наприклад, Presenter у моделі MVP [5]) та ін. Відповідні інструментальні засоби все ще знаходяться у стадії становлення. Як приклади з розряду вільно розповсюджуваних продуктів можна привести MediaWiki [6], Drupal [7], WordPress [8]. Неминучою розплатою для таких систем є прив'язка до серверних технологій, що обмежує можливості застосування в ситуаціях, коли серверне оточення для розробника фіксоване. При розгляді клієнтських бібліотек, що не залежать від серверних технологій, спостерігається їх спеціалізація: на маніпуляції з DOM-моделлю (jQuery [9], Zepto.js [10]), стильовому оформленні сторінок і елементах управління (Bootstrap [11], jQuery UI [12], w2ui [13]), побудові каркасів додатків (AngularJS [14], Backbone.js [15], Knockout [16]). При цьому використання бібліотек і модулів JavaScript дає величезні переваги при розробці веб-додатків. Серед очевидних плюсів – зменшення обсягу коду, підвищення зручності читання, підтримка кросбраузерності. Крім того, розробник отримує можливість правильно структурувати програмний код, тому що необхідність фізично змішувати HTML і JavaScript практично зникає. Незважаючи на багатий набір серед існуючих інструментальних засобів, залишаються завдання, вирішення яких актуальне при розробці

інформаційних систем: наявність диспетчера призначеного для користувача інтерфейсу, контроль цілісності даних при можливості доступу до системи багатьох користувачів.

Метою статті є створення архітектурного шаблона і методу побудови модулів для організації роботи віконної інформаційної web-системи на прикладі розробленого JavaScript-модуля, призначеного для організації бізнес-логіки при роботі з медичними сертифікатами.

Постановка проблеми: незважаючи на багатий набір існуючих інструментальних засобів, залишаються завдання, вирішення яких актуальне при розробці інформаційних web-орієнтованих систем: створення диспетчера інтерактивного інтерфейсу, призначеного для користувача, контроль цілісності даних при можливості доступу до даних системи багатьох користувачів.

Результати використання модульної архітектури показали, що розглянутий механізм створення диспетчера і віконного інтерактивного інтерфейсу web-форм не тільки органічно вписується в уже існуючі технології побудови web-додатків, але і сам має достатній потенціал, щоб у майбутньому стати ядром хмарних технологій розробки багатокористувацьких інформаційних систем і web-сервісів.

2. Розробка архітектури модуля

Модуль, який розроблюється, складається із двох відносно незалежних компонентів, що взаємодіють один з одним. Компоненти реалізовані у вигляді об'єктів JavaScript: об’єкт диспетчер вікон і клас віконного інтерактивного інтерфейсу. Скористаємося наявним у JavaScript механізмом доступу до об'єктів за допомогою оператора new, який використовується для створення об'єктів за допомогою функції власного конструктора, тим самим створюючи аналог класу. Такий конструктор зберігає екземпляр об'єкта в замиканні. Це дозволяє запобігати змінам об’єкта за межами функції-конструктора. При цьому використовується шаблон створення об’єктів – модуль і шаблон ізольованого простору імен [17].

Щоб без модифікації головних об'єктів модуля міняти не тільки зовнішній вигляд компонент форми, що відображаються, але і їх поведінку, при розробці застосовувався принцип проєктування, який відомий як шаблонний метод [18]. Код бібліотеки містить метод установки функцій зворотного виклику там, де розробнику необхідно доповнити стандартну обробку даних і подій своїми діями.

Механізм інтерактивного інтерфейсу, призначеного для користувача, крім функцій роботи з компонентами інтерфейсу, повинен відповідати таким вимогам:

• об’єкт web-форми повинен мати метод для відправки ajax-запиту на сервер і вміти обробляти відповіді сервера;

• об’єкт web-форми може бути як простим набором полів, так і містити підрозділи, вкладки або таблиці;

• зовнішній вигляд web-форм повинен мати можливість перенастроювання;

• на сторінці може бути кілька web-форм, які можуть взаємодіяти між собою.

Розроблений диспетчер вікон (ListWin) представляє собою JavaScript-об’єкт, який:

• зберігає колекцію згенерованих web-форм з їх компонентами та налаштуваннями;

• надає високорівневе API для маніпулювання web-формами і даними із клієнтських елементів управління;

• забезпечує взаємодію web-форми з DOM-моделлю web-документа;

• виконує попередню візуалізацію запущених процесів на формах;

• забезпечує для елементів управління механізм drag and drop.

Реалізація диспетчера ListWin складається із власного конструктора із privat-полями і public-методами. На рис. 1 розглянута об'єктна архітектура диспетчера.

```
var ListWin = function () {
    let Forms = [],      // массив форм
        numNewForm = 0,  // очередной уникальный идентификатор формы
        curObj = 0,      // номер текущего элемента (формы) (номер в массиве Forms)
        zIndex = 98;     // начальный уровень форм
    this.getDocumentHeight = function ( ){ ... }; // высота всей страницы
    this.getDocumentWidth = function( ){ …};  // ширина всей страницы
    this.createNewForm = function(arrProp){ ... }; // создание новой формы с параметрами
    this.getForm = function (num){ ... }; // возвращает ссылку на форму по её номеру
    this.setZIndex = function (curNum){ ... };   // перемещение на верхний уровень
    this.delForm = function ( ){ ... };  // удаление текущей формы
    this.setTimeoutWaitScreen = function (timeOut){ ... }; // закрытие текущей формы
    this.loadWait = function (modeL, modeS){ ... }; // прелоид выполнения операции
    this.emptyForm = function (cap, len){ ... }; // проверка существования формы
    this.mousedownDAD = function(object, funcMouseUp, hideClone) {...} // drag and drop
};

```

Рисунок 1 – Об’єктна архітектура диспетчера

JavaScript-об’єкт віконного інтерактивного інтерфейсу відповідає за функціонування додатку, обробку подій компонент-форм і включає в себе:

• методи рендеринга і маніпулювання web-формою: init(), changeCaption(), changeVisible(), setWidth(), setHeight(), WaitLoad();

• конструктор тригерів подій web-форми: initializationEvent();

• конструктор обробників подій елементів управління web-форми:addEvent().

Розглянемо програмну реалізацію форми-об’єкта віконного інтерфейсу (рис. 2). Форма розглядається як самостійний фрагмент призначеного для користувача інтерфейсу із власною логікою поведінки. Поняття форми за призначенням схоже з визначенням у специфікації HTML, але набагато функціональніше у способах реалізації і не обмежується DOM-елементом типу FORM. Функції об'єкта реалізовані у вигляді його public-методів.

```
var winObject= function () {
    let id = 0,       // идентификатор формы (главного контейнера div)
        idContainer;  // идентификатор контейнера формы
    this.collObj = {};
    this.WaitLoad = function(flag){ … }
    this.init = function (p_num, p_caption, p_visible){ … } // инициализация объекта
    this.changeCaption = function (p_name){ … };    // смена заголовка формы
    this.initializationEvent = function(){… }; // инициализация событий движения формы
    this.changeVisible = function (p_visible){ … }; // изменение видимости формы
    this.setWidth = function(w){ … }; // изменение ширины формы
    this.setHeight = function(h){ … };  // изменение высоты формы
    this.addObj = function(name_obj, arrProp){ … }; // добавление объекта с событиями
    this.addEvent = function(mask, event, strFunc){ … }; // назначение событий объектам
    this.wXHR = new XHRClass();
};
```

Рисунок 2 – Об’єкт віконного інтерфейсу

За допомогою об’єкта XHRClass реалізується транспортний рівень взаємодії клієнта з сервером, а саме завантаження даних для web-форм із сервера, збереження даних на сервер, асинхронні AJAX-запити до методів php-об’єктів сервера. Вся передача даних здійснюється у фоновому режимі без перезавантаження сторінки [19]. Для передачі структурованих даних між клієнтом і сервером використовується формат JSON.

Схема взаємодії компонент модуля представлена на рис. 3.

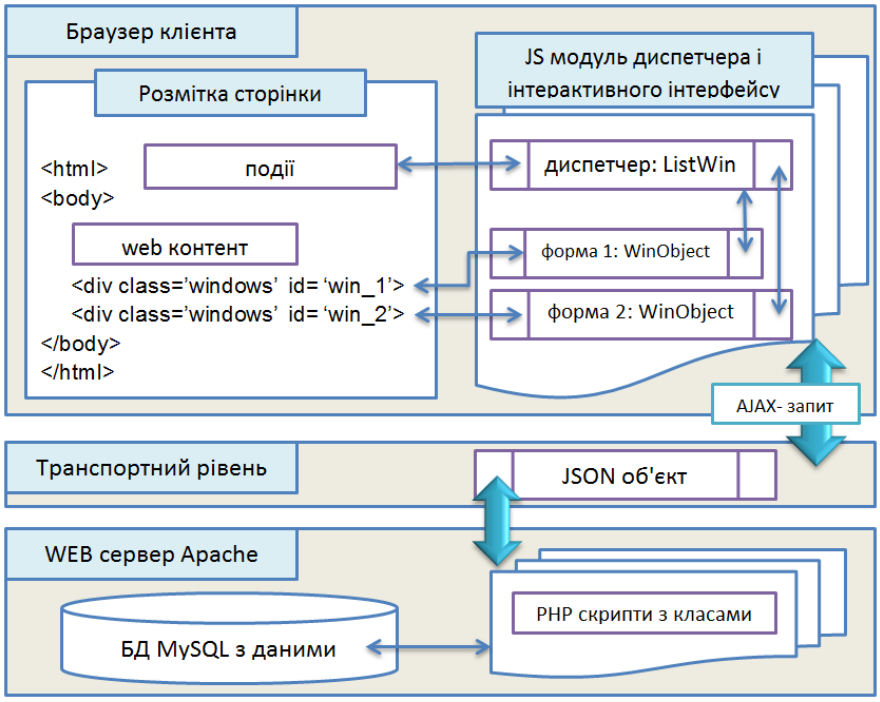

Рисунок 3 – Схема взаємодії компонент модуля

При конструюванні окремо взятого прикладного модуля на базі описаних об'єктів диспетчера і віконного інтерактивного інтерфейсу основним принципом створення об'єктів можуть виступати структурно-ієрархічні зв'язки. Такий підхід дозволяє оформити у вигляді прикладних модулів окремі інтерактивні інтерфейси.

На рівні взаємодії web-форм між собою структурні підходи вже не є достатньо ефективними, тому що лише мале число форм знаходиться в фіксованих відносинах виду «головний-підлеглий». Тому на даному рівні проведений перехід до мережевої моделі організації (рис. 3): форми win_1, win_2 є незалежними діючими об'єктами, що реагують на події своїх компонент за допомогою функцій зворотного виклику, які призначені при генеруванні інтерактивної форми диспетчером ListWin.

3. Побудова web-системи медичної сертифікації

Для організації хмарних обчислень і побудови web-системи на серверній стороні було використане програмне забезпечення:

- операційна система сервера – FreeBSD version 11.2;
- web-сервер Apache;
- препроцесор гіпертексту PHP version 7.4;
- система управління баз даних – MySQL server version 5.7.31.

Для роботи на стороні клієнта модулів програмного забезпечення web-системи сертифікації, розробленої на основі мови JavaScript і стандартів Standard ECMA-262, необхідна будь-яка операційна система, що підтримує www-навігатор з інтерпретатором мови JavaScript.

Застосуємо запропоновану модель і архітектурний шаблон до реалізації модуля формування та обробки медичних сертифікатів Modul_Sertifikat (рис. 4). За допомогою методу loadFormBaza() об'єкт забезпечує маніпуляції з DOM-моделлю web-документа. При цьому використовуються конструктор createNewForm() диспетчера ListWin і його метод addObj() для додавання елементів управління з обробниками подій.

```
var Modul_Sertifikat = {name : 'Modul_Sertifikat', formBaza : null,
                        arOrderBaza : {order : 'FIO', dir : 'ASC'},
                        arSymbolDir : {'DESC' : '&#9650;', 'ASC' : '&#9660;'} };
Modul_Sertifikat.loadFormBaza = function(){
let optOglad = this.optionOfArray(this.arOglad);
     this.formBaza = ListWin.createNewForm({caption : 'Сертифікати про проходження профілактичного огляду',
                                            height: 235, width : 880, visible : 1, WaitScreen : 0});
     if (this.formBaza) with (this.formBaza){
        addObj('div', {data : 'ПІП: ', pos : [439, 38], style:'font-size:14pt; color:#132F76;'});
        addObj('input', {id : 'FIO', type:'text', pos : [480, 35], wh : [288, 22], class : 'defaultInp',
                         style : 'font-size:10pt; border:2px solid #92C8ED;',
                         events :{oninput : this.name+'.inputData(this, true);' } });
        addObj('table', {id : 'kolontitul_tabl', data : kolontitul_tabl,  class : 'tab_class', wh : [835, 30],
                        style : 'margin-left:-2px; margin-top:88px; border: solid 2px #759cdd; '} );
        addObj('div', {id : 'containerTabl', wh : [835, 460], pos : [13, 165],
                       style : 'overflow: hidden auto; border: solid 2px #759cdd; visibility: hidden;'} );
        addObj('table', {id : 'tabl', parent : 'containerTabl', data : '<tbody></tbody>', class : 'tab_class',
                        style : 'position: relative; width:100%;' } );
        addObj('img', {id : 'WaitLoad', src: "img/loader.gif", wh : [70, 70], style : 'left:45%; top:25%;'});
        addObj('button', {data : '<span style="font-weight:bold; ">&#9672; Додати нового клієнта</span>',
                          class : 'buttonform', style : 'right:190px; height:50px; width:145px;',
                          id : 'buttonIn', events : {onmouseup : this.name+'.cartaPacient();'} });
        addObj('button', {data : 'Закрити', class : 'buttonform',  style : 'right:20px; font-size: 12pt;',
                          events : {onmouseup : 'onClick_CloseForm()'} });
     }
     Modul_Sertifikat.refreshBaza();
};
```

Рисунок 4 – Модуль обробки медичних сертифікатів

Після генерації форми викликається метод refreshBaza() (рис. 5). У методі ajax-запит направляється на сервер скрипта getSertifikat.php у метод listFIO класу, який готує необхідні дані і організовує логіку для клієнтської сторони. Підготовлена інформація в JSON-об'єкті доставляється у клієнтський браузер, де за допомогою анонімної функції зворотного виклику перетворюється в параметри елементів управління. Доступ до елементів форми виконується через колекцію об'єктів форми collObj.

```
Modul_Sertifikat.refreshBaza = function(){
let kodMed = this.formBaza.collObj['selectMed'].value;
    this.formBaza.WaitLoad(true);
    this.formBaza.wXHR.query({param:'listFIO', idustanova : kodMed,
                      FIO : formBaza.collObj['FIO'].value,
                      order : this.arOrderBaza.order, dir : this.arOrderBaza.dir}, 'php/getSertifikat.php');
    this.formBaza.wXHR.request.onreadystatechange = function(){
        if (this.readyState == 4 && this.status == 200){
            Modul_Sertifikat.formBaza.setHeight(700);
            Modul_Sertifikat.formBaza.WaitLoad(false);
            Modul_Sertifikat.formBaza.collObj['containerTabl'].style.visibility = 'visible';
            answer = this.responseText.split('|');
            Modul_Sertifikat.formBaza.collObj['tabl'].querySelector('tbody').innerHTML = answer[0];
            System.changeWidthTabl(formBaza.collObj['containerTabl'], formBaza.collObj['tabl'], 850, 831);
        }
    }
};
```

Рисунок 5 – Метод об’єкта для запиту даних на сервері

На рис. 6 показаний код класу, який отримує параметр param із запиту request і виконує всю бізнес-логіку на стороні сервера.

```
<?php
    header("Content-type: text/html; charset=UTF-8");
    include_once "Lists.php";
    include_once "IOAction.php";

    $objSertifikat = new clSertifikat();
    $funcName = $_POST['param'];
    echo $objSertifikat->$funcName();

Class clSertifikat{
    public    $modulJS = 'ModulSertifikat';
    private   $system = 'MedSystem';
    protected $title = 'Модуль для роботи з сертифікатами';

    // КОНСТРУКТОР КЛАССА С НАЧАЛЬНОЙ ИНИЦИАЛИЗАЦИЕЙ
    public function __construct($p1)...//
    function __destruct()...//
    // конвертирование строки
    public function convert($text, $move = 'cp1251')...//
    // формирование таблицы базы клиентов с сертификатами
    public function listFIO()...//
    // получение данных карты
    public function dataKarta()...//
    // удаление карты клиента
    public function delCarta()...//
}
?>
```

Створення екземпляра класу. Звернення до методу, зазначеному в пакеті AJAX запиту

Реалізація php класу роботи з сертифікатами з заданим функціоналом

Рисунок 6 – Клас сервера для обробки запитів клієнта

На рис. 7 представлений результат генерації форми бази сертифікатів із заповненими даними. Отриманий із боку сервера JSON-об’єкт був перетворений у параметри форми: у дворівневу таблицю із клієнтською базою та списком сертифікатів, що належать окремим особам; у призначені функції зворотного виклику для подій контекстного меню вибору клієнтів і роботи з сертифікатами; в загальну інформацію про кількість сертифікатів по заданому фільтру. При виборі пункту редагування карти клієнта генерується нова форма з ajax-завантаженням даних клієнта з сервера.

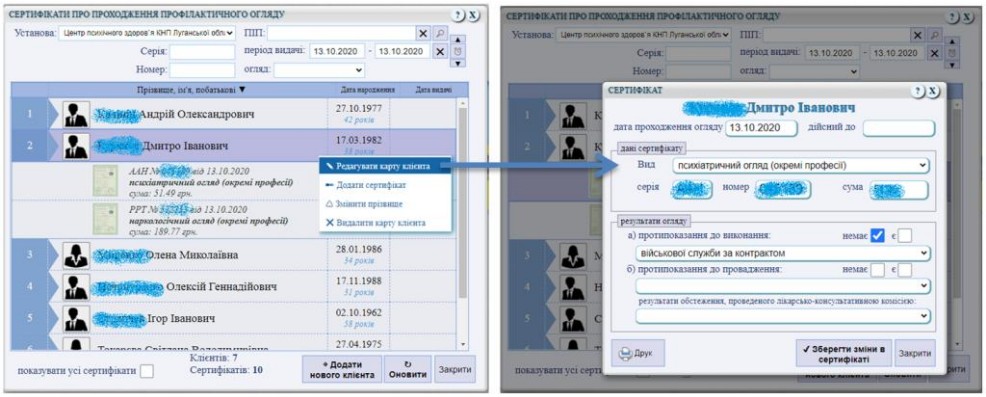

Рисунок 7 – Генерація форми бази сертифікатів

Окремі пункти при формуванні сертифіката, такі як «Результати огляду», асинхронно завантажуються з бази даних сервера при створенні форми. Для їх зміни і редагування в системі створюються окремі інтерфейсні вікна (рис. 8). При необхідності сформований сертифікат (рис. 8) може бути збережений у форматі pdf або роздрукований на принтері.

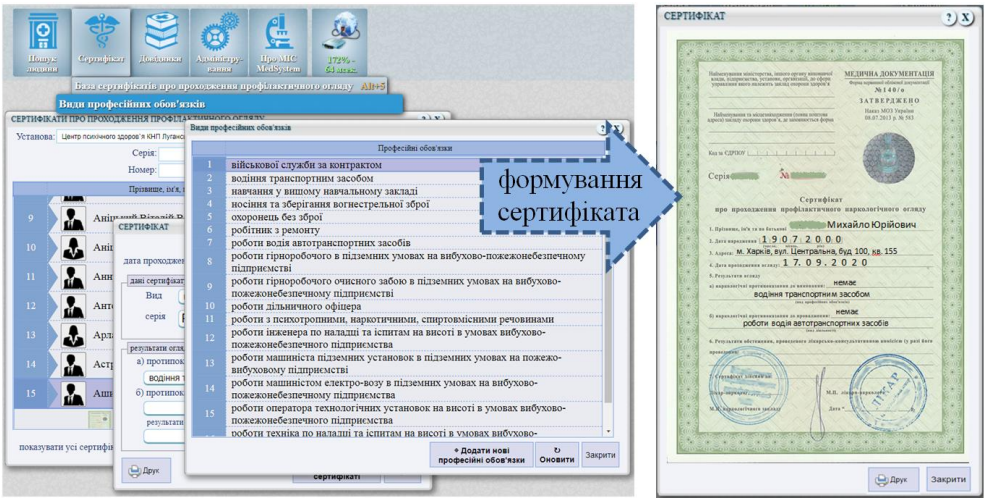

Рисунок 8 – Сформований сертифікат

4. Висновки

Запропонований у роботі шаблон архітектури модуля для реалізації диспетчера та функціонала віконного інтерфейсу користувача дозволив створити модуль інформаційної системи для організації бізнес-логіки при роботі з медичними сертифікатами. Модульний підхід позитивно позначився на швидкості призначеного для користувача інтерфейсу і можливості масштабування функціонала самої системи при реалізації хмарних технологій. Завдяки модульному підходу, час рендеринга web-форм із вихідними даними в більшості випадків є дуже незначним у порівнянні з часом, необхідним браузеру для розбору і відображення всього масиву даних бази, що позитивно позначається на швидкості призначеного для ко-

ристувача інтерфейсу. Величезне практичне значення має генерація web-форм, тому що налагодження помилок у цій програмній структурі є болючим місцем багатьох фреймворків.

Результати використання системи показали, що механізм модульного створення диспетчера і віконного інтерактивного інтерфейсу web-форм не тільки органічно вписується в уже існуючі технології побудови web-додатків, але і сам має достатній потенціал, щоб у майбутньому стати ядром хмарних технологій розробки багатокористувацьких інформаційних систем і web-сервісів.

СПИСОК ДЖЕРЕЛ

1. Медведєв А. Хмарні технології: тенденції розвитку, приклади виконання. *Сучасні технології автоматизації*. 2013. № 2. С. 6–9.
2. HTML 4.01 Specification. URL: https://www.w3.org/TR/html401/ (дата звернення: 15.10.2020).
3. Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. URL: https://www.w3.org/TR/CSS22/ (дата звернення: 15.10.2020).
4. ECMAScript Language Specification – ECMA-262 Edition 5.1. URL: http://www.ecma-international.org/ecma-262/5.1/ (дата звернення: 15.10.2020).
5. Архітектура MVP. URL: http://www.gwtproject.org/articles/mvp-architecture.html (дата звернення: 15.10.2020).
6. MediaWiki. URL: http://www.mediawiki.org/wiki/MediaWiki (дата звернення: 15.10.2020).
7. Drupal – Open Source CMS, drupal.org. URL: https://drupal.org/ (дата звернення: 15.10.2020).
8. WordPress. Русский. URL: http://ru.wordpress.org/ (дата звернення: 15.10.2020).
9. jQuery. URL: http://jquery.com/ (дата звернення: 15.10.2020).
10. Zepto.js: the aerogel-weight jQuery-compatible JavaScript library. URL: http://zeptojs.com/ (дата звернення: 15.10.2020).
11. Bootstrap. URL: http://getbootstrap.com/ (дата звернення: 15.10.2020).
12. jQuery UI. URL: http://jqueryui.com/ (дата звернення: 15.10.2020).
13. w2ui: Home. JavaScript UI. URL: http://w2ui.com/web/ (дата звернення: 15.10.2020).
14. AngularJS – Superheroic JavaScript MVW Framework. URL: http://angularjs.org/ (дата звернення: 15.10.2020).
15. Backbone.js. URL: http://backbonejs.org/ (дата звернення: 15.10.2020).
16. Knockout: Home. URL: http://knockoutjs.com/ (дата звернення: 15.10.2020).
17. Стефанов С. Javascript. Шаблони. Санкт-Петербург, 2011. 263 с.
18. Гамма Е., Хелм Р., Джонсон Р., Вліссідес Дж. Прийоми об'єктно-орієнтованого проектування. Патерни проектування. Санкт-Петербург, 2001. 368 с.
19. Крейн Д., Паскарелло Э., Джеймс Д. AJAX в действии. М.: Вильямс, 2006. 640 с.

Стаття надійшла до редакції 19.10.2020