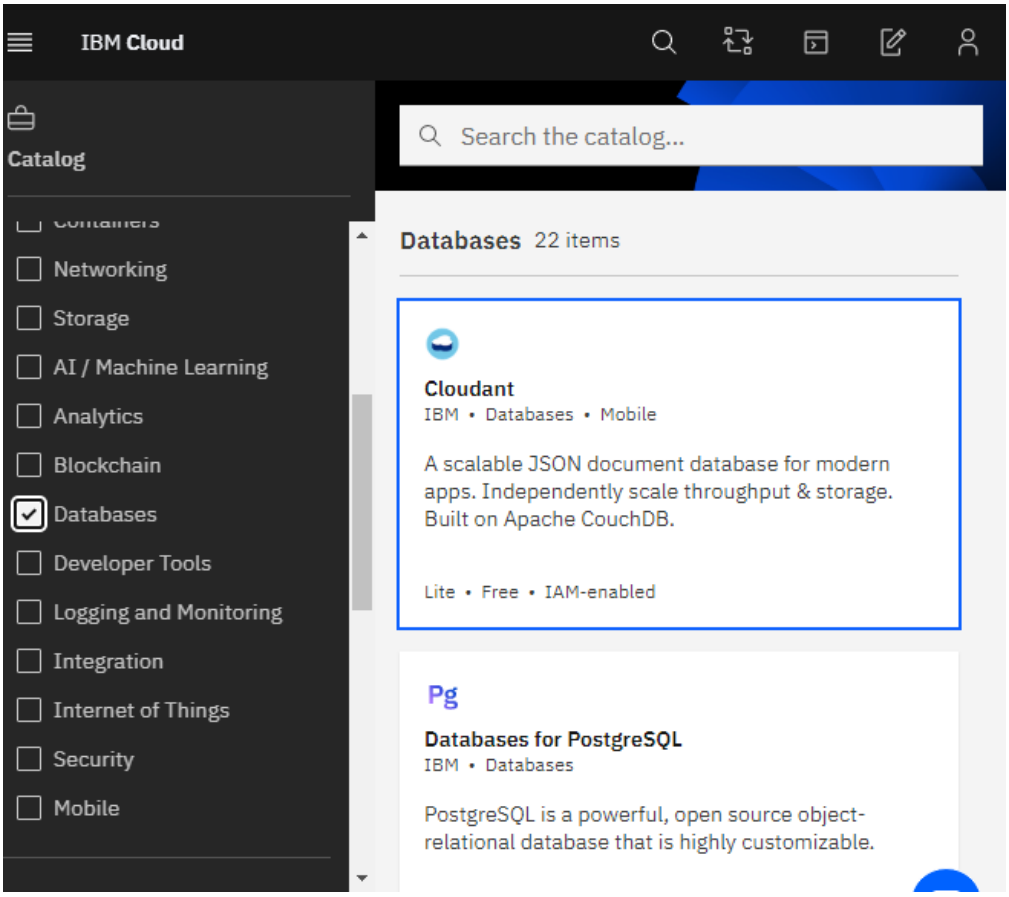
IBM Cloudant

I. Въведение

IBM Cloudant е NoSQL документен тип база данни. Достъпът до тази база данни се реализира чрез протокол HTTPS. Има опция да използвате IBM Cloudant безплатно. За целта трябва да се регистрирате като клиент на IBM Cloud. След като се регистрирате, запомнете своята парола. След успешна авторизация на достъпа (login) създайте нов ресурс. Натиснете бутон “Create resource”, изберете “Databases”, а след това Cloudant. Изберете безплатен абонаментен план и задайте име на ресурса.



Фиг. 1 Избор на ресурс Cloudant

Попълнете полетата от меню “Create”. Изберете “Frankfurt” или “London” за “Available regions”. За поле “Authentication method” изберете “IAM and legacy credentials”. Този избор ще позволи отдалечен достъпа до базите данни чрез базова авторизация. И накрая, изберете безплатен достъп до услугата – план “Lite”. При този план ограниченията са следните:

- 20 четения/сек.
- 10 записа/сек.
- 5 глобални заявки/сек.
- 1 GB място за съхранение

За да активирате услугата натиснете бутон “Create”. При успешно активиране ще бъдете препратени към страницата с вашите ресурси. Изчакайте докато статуса на услугата стане “”Active”.

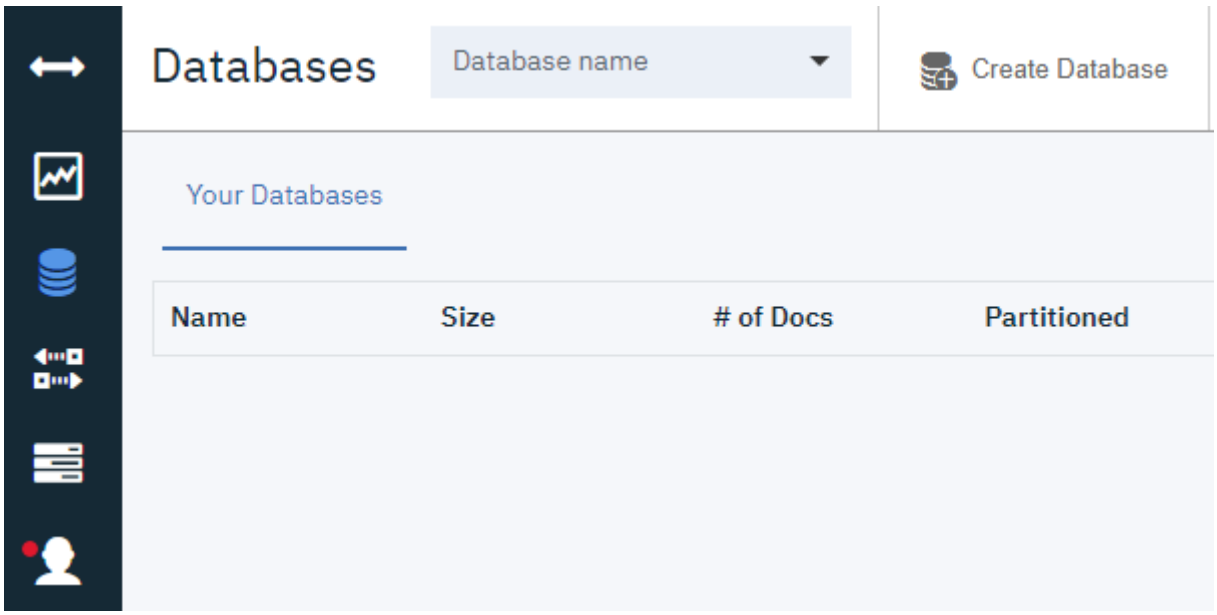
Изберете услугата, за да достигнете до менюто за настройки. Изберете “Manage”. Копирайте URL низа, съответстващ за “External Endpoint (preferred)”, например:

https://71176003-6559-48ff-95d9-9ff0384b35d7-bluemix.cloudantnosqldb.appdomain.cloud

Чрез този адрес ще достъпвате вашата база данни отдалечено чрез HTTPS клиент.

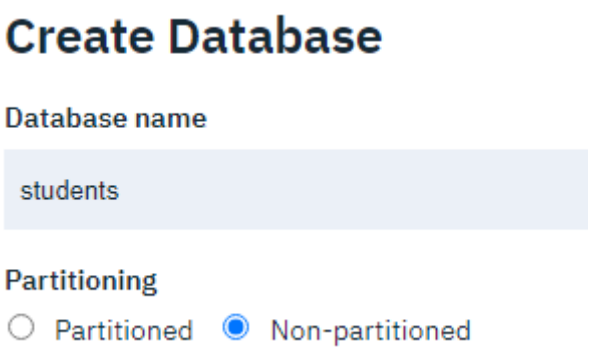
Изберете “Service credentials” и натиснете бутон <New credential +>. Задайте име на удостоверяването и “Manager” за роля. Натиснете бутон <Add>.

Натиснете бутон <Launch Dashboard>. Така ще достигнете до графичния интерфейс на IBM Cloudant чрез който може да управлявате своите бази данни (виж Фиг. 2).



Фиг. 2 IBM Cloudant dashboard

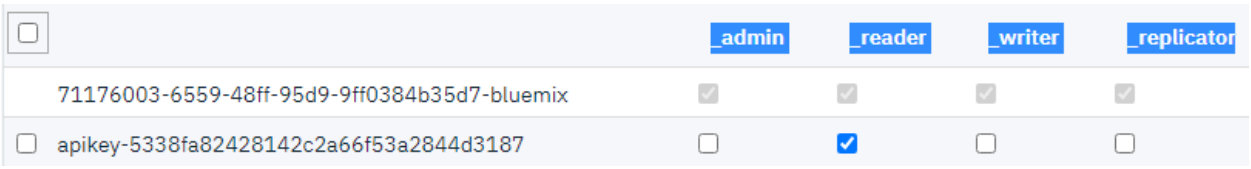
Вече сте готови да създадете своята първа база данни. За целта натиснете бутон <Create Database>. Изберете име на базата данни, например “students” и забранете разделянето на базата данни между множество нодове (сървъри) – “Non partitioned”.



Фиг. 3 Създаване на нова база данни

Натиснете бутон <Create>. Вече имате база данни, но в нея все още няма документи. Преди да създадем необходимите документи ще зададем необходимите разрешения за

достъп до услугата. За целта от менюто изберете “Permissions” и натиснете бутон <Generate API Key>. Ключовете са необходими, за да се предостави програмен достъп до базите данни на Cloudant. Можете да генерирате толкова ключа, колкото са ви необходими, например: един за администратора на базите данни, един или повече за четене и запис и един или няколко само за четене. Системата ще генерира двойка “ключ-стойност”. Трябва да го копирате и запишете, за да може да го използвате на по-късен етап. За всяка двойка “ключ-стойност” може да зададете различни права за достъп (виж Фиг. 4).



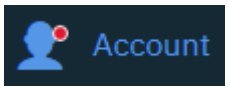
Фиг. 4 Задаване на права за достъп (permissions)

За да въведете документ в базата данни “students” натиснете бутон <Create document>. Документът ви автоматично получава идентификатор чрез системното поле “_id”. Може да зададете нова стойност на полето, но то трябва да е уникално спрямо останалите документи. Нека да въведем в документа информация за студентите от MongoDB базата данни “students”, колекция “data1”. Чрез JSON редактора на Cloudant въведете информацията за един студент (виж Фиг. 5). Натиснете бутон <Create Document> с което ще запишете документа в база данни “students”.

```
{
  "_id": "student-0001",
  "id": "21705001",
  "name": {
    "firstName": "Георги", "lastName": "Котев"
  },
  "grade": [
    {
      "subject": "НБД", "value": 4
    },
    {
      "subject": "КМС", "value": 2
    },
    {
      "subject": "ММС", "value": 4
    },
    {
      "subject": "ДСП", "value": 3
    }
  ]
}
```

Фиг. 5 Създаване на нов документ

След като запишете документа, той получава още едно системно поле “_rev”. То съдържа информация за версията на документа и не трябва да бъде манипулирано. Това поле се използва за системни нужди (репликация на базата данни). По аналогичен начин създайте още два записа в базата данни. Вече имате база данни с три документа в нея. Остава да разрешим отдалечения достъп до тази база данни. За целта от вертикалното меню изберете “Account”:

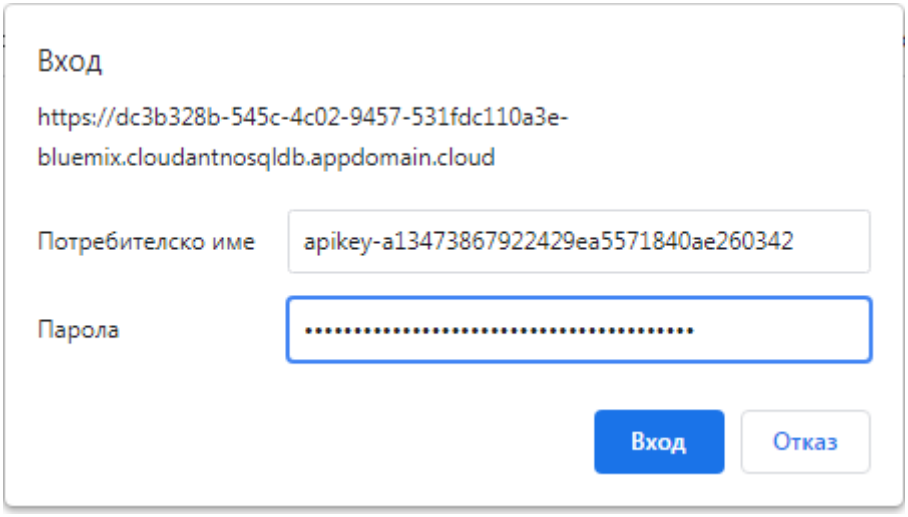


От хоризонталното меню изберете Cross-Origin Resource Sharing (CORS). Споделянето на ресурси от различни източници (CORS) ви позволява да се свързвате с отдалечени сървъри директно от браузъра. Натиснете бутон <Enable CORS>. От “Origin domains” изберете “All Domains (*)”. По този начин достъпът до вашите бази данни ще е разрешен от всеки IP адрес. Защитата на връзката се реализира чрез протокол HTTPS и избрания механизъм за автентикация на достъпа. По подразбиране се използва базова автентикация при която ключа и паролата се кодират чрез алгоритъм Base64.

Тествайте достъпа до услугата като от браузъра направете заявка до един от документите в база данни “students”, например:

https://dc3b328b-545c-4c02-9457-531fdc110a3e-bluemix.cloudantnosqldb.appdomain.cloud/students/student-0001

Тъй като достъпът изисква базова авторизация чрез име и парола, браузърът ще визуализира прозорец за тяхното въвеждане (виж Фиг. 6).

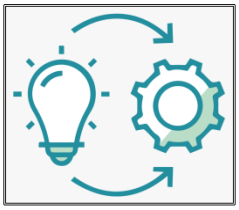


Фиг. 6 Форма за авторизация на достъпа

След натискане на бутон <Вход> се визуализира съдържанието на избрания документ:


```
{"_id":"student-0001","_rev":"1-
9c8f87b7104684e21c832ffae329fd69","id":"21705001","name":
{"firstName":"Георги","lastName":"Котев"},"grade":
[{"subject":"НБД","value":4},{"subject":"КМС","value":1},
{"subject":"ММС","value":1},{"subject":"ДСП","value":1}]}
```

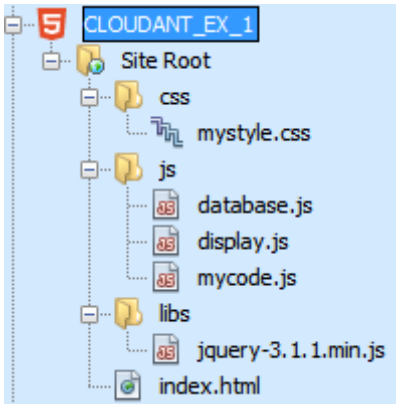
II. Задачи за изпълнение



Задача 1: *Направете своя регистрация в IBM Cloud. Следвайте стъпките за получаване на достъп до ресурса IBM Cloudant. Създайте база данни "students" и запишете поне три документа, които описват студенти и техните оценки.*

Задача 2: *Напишете Web приложение, което реализира връзка с IBM Cloudant, база данни "students" и визуализира информация за всеки студент за който има запис в базата данни.*

Създайте нов HTML5 NetBeans проект с име CLOUDANT_EX_1. Структурата на приложението е показана на Фиг. 7.



Фиг. 7 Структура на проекта – Задача 2

Индексният файл (**index.html**) съдържа HTML5 кода чрез който се изгражда изгледа на приложението. Информацията, което се визуализира е следната:

Горен колонтитул (header)
Информация В даден момент от време се визуализира съдържанието на една от четири възможни секции.
Долен колонтитул (footer)

За да реализираме SPA приложение (с една HTML5 страница) ще използваме няколко HTML секции, като в даден момент ще визуализираме само една от тях (другите ще бъдат скрити). На Фиг. 8 е показано съдържанието на файл index.html. За да използваме кирилица сме задали да бъде активна кодова таблица UTF-8. Заредени са и всички JavaScript и CSS модули, необходими за работа на приложението. Информацията, която се визуализира е в тялото на етикет <main>. Тя е разпределена в 4 секции със следното предназначение:

- Секция "loading-box" – показва низа "Свързване…" при всяко начало на мрежов обмен.
- Секция "query-box" – показва бутон чрез който се реализира изпращане на заявка към IBM Cloudant.
- Секция "info-box" – показва форматираното съдържание на получените документи.

- Секция “error-box” – показва информация за грешката, ако такава възникне.

```
<!DOCTYPE html>
<html>
    <head>
        <title>IBM Cloudant</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width,
                                       initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href="css/mystyle.css"/>
        <script src="libs/jquery-3.1.1.min.js"></script>
        <script src="js/database.js"></script>
        <script src="js/display.js"></script>
        <script src="js/mycode.js"></script>
    </head>
    <body>
        <header>IBM Cloudant Query</header>
        <main>
            <section data-position="loading-box">
                Свързване ...
            </section>
            <section data-position="query-box">
                <button id="submit">Изпрати заявка</button>
            </section>
            <section data-position="info-box">
            </section>
            <section data-position="error-box">
                <div id="error-info"></div>
            </section>
        </main>
        <footer>&copy; 2021 rs-soft-bg</footer>
    </body>
</html>
```

Фиг. 8 Програмен код на index.html

Останалият програмен код е структуриран в няколко папки. В папка “**libs**” са библиотеките, необходими за реализация на проекта. На този етап ще използваме само библиотека jQuery. В папка “js” са записани три JavaScript модула:

- **mycode.js** – съдържа програмния код чрез който се изчаква зареждане на библиотека jQuery, прехваща се събитието, което се генерира при натискане на бутона за изпращане на заявка, прехващат и събитията “Начало на AJAX заявка” и “Край на AJAX заявка”. Тези събития се използват от интерфейса с потребителя - уведомяват клиента за всяко начало и край на AJAX заявки. Те се използват с цел адресиране на ресурси от IBM Cloudant.
- **display.js** – съдържа програмния код чрез който се реализира SPA приложение. Предвидени са функции за показване и скриване на съдържание от различни секции от HTML5 кода.
- **database.js** – съдържа програмния код за изпращане на заявка къмIBM Cloudant, получаване на отговора и динамично формиране на отговор за браузъра.

В папка “**css**” е CSS кода, който реализира желаното стилово форматиране.

Програмният код от модул “**mycode.js**” е показан на Фиг. 9. След зареждане на библиотека jQuery - $(function() { ... }); - се прехваща три събития: “click” от бутона за изпращане на заявка и “ajaxStart” и “ajaxStop” от библиотека jQuery. При натискане на бутона управлението се предава на функция **sendQuery** от модул “database.js”. Веднага след зареждане на jQuery се визиализира секция “query-box” чрез метод **view** (модул display.js).

```
$(function () {
    $("#submit").click(function() {
        sendQuery();
    });
    $(document).on({
        ajaxStart: function () {
            view('loading-box');
        },
        ajaxStop: function () {
            hide('loading-box');
        }
    });
    view('query-box');
});
```

Фиг. 9 Програмен код на mycode.js

```
function view(viewName) {
    $('main > section').hide();
    $(`[data-position=${viewName}]`).show();
}
function hide(viewName) {
    $(`[data-position=${viewName}`).hide();
}
function showInfo(data) {
    $('[data-position=info-box]').append(data);
    view('info-box');
}
function showError(errorMsg, time) {
    var errorBox = $('#error-info');
    errorBox.text(errorMsg);
    view('error-box');
    setTimeout(function () {
        $('[data-position=error-box]').fadeOut();
        view('query-box');
    }, time*1000);
}
```

Фиг. 10 Програмен код на display.js

На Фиг. 10 е показан програмния код от файл “**display.js**”. Функция **view** скрива всички секции, които са в тялото на етикет <main> и показва само секцията, чието име се предава като аргумент. Функция **hide** скрива съдържанието на указана по име секция. Информацията, която е получена от базата данни, се визуализира чрез функция **showInfo**. Тази функция се вика толкова пъти, колкото документа има в базата данни. Затова тя добавя всеки път ново HTML съдържание към секцията чрез функция **append**. Обработката на грешки се реализира от тялото на функция **showError**. Тя получава два аргумента: низ, който описва грешката “errorMsg” и времето в секунди за което този низ да се вижда на екрана “time”.

Програмният код, който реализира комуникация с базата данни IBM Cloudant е в модул **database.js**. Тази услуга в облака на IBM ни позволява да пишем Web приложения без реален код от страна на сървъра (serverless app). Комуникацията между клиента и базата данни е чрез протокол HTTPS. Целият трафик е шифриран чрез SSL/TLS. Тъй като ще използваме базова авторизация на достъпа до Cloudant е необходимо да кодираме ключа и паролата за достъп до база данни “students” чрез алгоритъм Base64. В JavaScript това се реализира чрез вградената функция **btoa**. Ще използваме библиотека jQuery за да изпратим заявката като AJAX заявки. За целта ще използваме функция **ajax**. За да можем

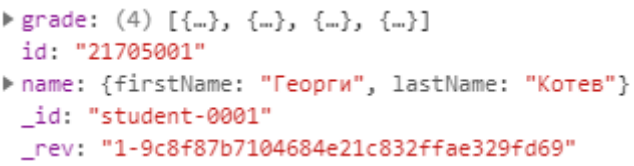
да зададем timeout интервал трябва да използваме асинхронния вариант на тази функция. Това означава, че функцията веднага ще върне от управление след изпълнението ѝ. За да разберем кога реално е получен отговор ще използваме обект-обещание (promise). Функцията, която реализира комуникация с IBM Cloudant, е с име **getResult**. Тя се вика от функция **sendQuery**. Синхронизацията между тези две функции се реализира чрез обекта-обещание. Когато се получи валиден резултат от услугата се вика анонимна функция (**then**). Грешките се обработват също чрез анонимна функция (**catch**). Ако резултатът е валиден, но се съдържа само един документ, неговото съдържание се парсва и добавя в HTML страницата чрез функция **addResultToPage**. Ако резултатът съдържа множество JSON обекти, те се обработват последователно (вътрешен цикъл чрез функция **forEach**) отново със същия метод.

Когато заявяваме съдържание на документ от Cloudant база данни трябва да използваме следния URL низ:

```
const url = PROTOCOL + SERVER_NAME + ':' + PORT + '/' + DATABASE + '/' +
            DOCUMENT + REC_FIELDS;
```

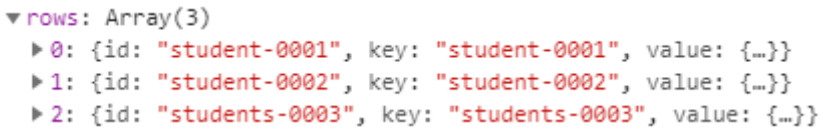
където:

- PROTOCOL е низа ‘https://’.
- SERVER_NAME е името на сървъра (получавате го при инсталиране на услугата).
- PORT – номера на порта, който Claudant подслушва – 443 (HTTPS).
- DATABASE – име на базата данни, до която желаете достъп – “students”.
- DOCUMENT – име на документа до който желаете достъп. В този случай резултатът ще бъде съдържанието на указания документ:

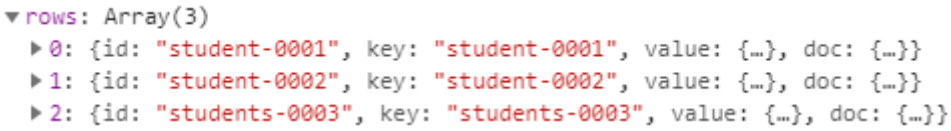


Ако трябва да получите имената или съдържанието на всички документи, стойността на това поле трябва да бъде **“_all_docs**”.

- RES_PARS – допълнителни параметри, които се предават към заявката. Ако полето е празен низ, при DOCUMENT=“_all_docs” ще получите масив от JSON обекти, които чрез поле “id” описват имената на всеки един документ, например:



Ако искате с една заявка да получите и съдържанието на документите, то задайте за стойност на RES_PARS низа “ ?include_docs=true”:



Ако към документите има прикачени файлове и искате да получите и тяхното съдържание трябва да изпратите втори параметър с име “attachments” със стойност true:

“&attachments=true”.

Припомнете си за какво служеха специални символи “?” и “&” при изпращане на HTTP заявки.

На Фиг. 11 са показани константите, които модул database.js използва. Не забравяйте да смените стойностите на константите SERVER_NAME, KRY и PASWWORD с тези за вашата база данни. Времето за timeout се задава в ms и следователно е 4 секунди.

```
const PORT = 443;
const TIMEOUT = 4000;
const PROTOCOL = 'https://';
const SERVER_NAME =
 'dc3b328b-545c-4c02-9457-531fdc110a3e-
bluemix.cloudantnosqldb.appdomain.cloud';
const DATABASE = 'students';
const DOCUMENT = '_all_docs';              // 'student-0001'
const RES_PARS = '?include_docs=true';     // ''
const KEY = 'apikey-a13473867922429ea5571840ae260342';
const PASSWORD = '53a874f75d9825ff9aa9c10d3c6ace4449da2045';
```

Фиг. 11 Използвани константи – database.js

Програмният код от функция **getResult** е показан на Фиг.12.

```
function getResult(hash) {
    var url = PROTOCOL + SERVER_NAME + ':' + PORT + '/'
            + DATABASE + '/' + DOCUMENT + RES_PARS;
    var promise = new Promise(function (resolve, reject) {
        $.ajax({
            type: 'GET',
            url: url,
            async: true,
            cache: false,
            contentType: 'application/json',
            timeout: TIMEOUT,
            beforeSend: function (req) {
                req.setRequestHeader('Accept', 'application/json');
                req.setRequestHeader('Authorization', 'Basic ' + hash);
            },
            success: function (data) {
                resolve(data);
            },
            error: function (data) {
                reject(data);
            }
        });
    });
    return promise;
}
```

Фиг. 12 Функция getResult

Функция **ajax** от библиотека jQuery изисква един атрибут – JSON обект който съдържа множество полета чрез които се указва каква точно заявка искаме да изпратим. Използван е HTTP метод GET (получаване на съдържание). Заявките не трябва да се кешират (cache: false). Задължително трябва да е разрешено асинхронното изпълнение на заявката (async: true), за да можем да зададем timeout интервал. Задайте за тип на отговора MIME типа на JSON документ (application/json). За да зададем използването на базова авторизация ще използваме функция **beforeSend** и чрез функция

setRequestHeader ще вмъкнем в заглавния блок на заявката поле “Authorization” със стойност “Basic xxx”, където “xxx” е Base64 кодиран низ, който съдържа “ключа + символ **:** + паролата”.

При успешно изпълнение на заявката се вика функция **success**. От тялото ѝ се вика функция **resolve**, която предава управлението на **then**. При грешка се вика функция **error**. От тялото ѝ се изпълнява функция **reject** и управлението се предава на **catch**.

Програмният код от функция **sendQuery** е показан на Фиг. 13.

```
function sendQuery() {
   var hash = btoa(KEY + ':' + PASSWORD);
   getResult(hash)
      .then(function (result) {
         var n = Object.keys(result).length;
         if (n === 1) {
            addResultToPage(result);
         }
         else {
            var rows = result.rows;
            rows.forEach(function(row) {
               addResultToPage(row.doc);
            });
         }
      }).catch(
         function (error) {
            var status = error.status;
            if (status === 0) {
               showError('Невъзможна е връзката с базата данни!', 3);
            }
            else if (status === 401) {
               showError('Грешка при авторизация!', 3);
            }
         }
   );
}
```

Фиг. 13 Функция sendQuery

Функция **sendQuery** вика функция **getResult** и синхронизира завършването ѝ (успешно или неуспешно) чрез обект-обещание. За да проверим резултата дали съдържа един или множество JSON обекта използваме функция **keys** и свойство “length”. Ако стойността на променлива *n* е 1, това означава, че сме заявили съдържанието на точно определен документ. В противен случай сме заявили съдържанието на всички документи. Обработката на резултата и в двата случая се реализира чрез функция addResultToPage.

Програмният код от функция **addResultToPage** е показан на Фиг. 14. Функцията извлича името на студента и факултетния му номер. След това вмъква тази информация в div контейнер с идентификатор “info-name”. След това информацията за оценките се извлича и вмъква в контейнер с идентификатор “info-grades”. Чрез функция **showInfo** тази информация се добавя към секция “info-box”.

```
function addResultToPage(result) {
    var id = result.id;
    var firstName = result.name.firstName;
    var lastName = result.name.lastName;
    var name = `${firstName} ${lastName} - ${id}`;
    var divName = '<div id="info-name">' + name + '</div>';
    var grades = result.grade;
    var gradesHtmlStart = '<ul class="grades">';
    var gradesHtmlEnd = '</ul>';
    var gradesHtml = gradesHtmlStart;
    grades.forEach(function (grade) {
        var subject = grade.subject;
        var value = grade.value;
        var pair = `${subject}: ${value}`;
        gradesHtml += `<li class="grades">${pair}</li>`;
    });
    gradesHtml += gradesHtmlEnd;
    var divGrades = '<div id="info-grades">' + gradesHtml + '</div>';
    showInfo(divName + divGrades);
}
```

Фиг. 14 Функция addResultToPage

На Фиг. 15 са показани снимки от потребителския интерфейс на приложението.

Задача 2: *Разширете функциите на приложението от Задача 1 така, че да се дава възможност за търсене на студентите, които имат конкретна оценка по избрана дисциплина.*

Фиг. 15 Потребителски интерфейс: a) Начална страница; b) Липсва мрежова свързаност; c) Резултат при успешно изпълнена заявка