

# FEEDBACK FOR SMART SCHOOLS

Titrat Cristina Georgiana

Clasa a XI-a

Profesor: Anca Adina

Liceul Teoretic “Neagoe Basarab”,  
Oltenița

# Cuprins

Tehnologiile folosite .....	3
Aplicații folosite .....	4
Tema proiectului .....	4
Structura proiectului .....	5
Prezentarea proiectului .....	6
Indicații de instalare a proiectului.....	17
Bibliografie .....	19

# Tehnologiile folosite

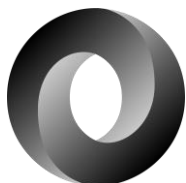
- <HTML5>
- .CSS3
- JavaScript.js
- {{AngularJS}}
- require("Node.js");
- connect(„MongoDB")
- "data": JSON[]
- Bootstrap
- \$(jQuery)
- Socket.io



socket.io



express



JS

**HTML**



**CSS**



mongoDB



# Aplicațiile folosite

Pentru realizarea proiectului am utilizat:

- Editoarele de cod Brackets și Sublime
- MongoDB Compass Community

## Tema proiectului

Îmi doresc ca școlile să asculte opinia publică și să se raporteze la cerințele actuale, fiind înconjurați de tehnologie, manualele devin din ce în ce mai neatractive, iar elevii încet, încet își vor pierde atenția, dacă nu se va întâmpla o schimbare.

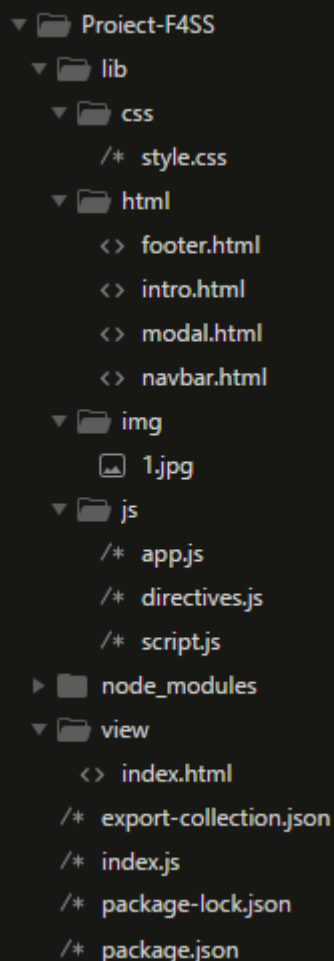
Dar cum sa apară o schimbare, când licee nu știu ce își doresc elevii de la școală?

Consider că opinia publică este cea mai importantă, astfel ascultând cerințele utilizatorilor (părinții, actuali sau foști elevi, chiar și bunici) putem afla mai ușor ce își doresc de la școală și ce vor să se îmbunătățească.

Am dorit să realizez o platformă web pentru opinii, numită „Feedback for smart schools” care se adresează atât liceelor cât și elevilor.

Scopul paginii este de a ajuta dezvoltarea liceelor în procesul de modernizare, prin ascultarea opiniilor, neregulilor sau sesizărilor. Aceasta se adresează și viitorilor elevi de liceu care vor să afle mai multe informații, pentru a face o alegere cât mai bună în alegerea viitoarei instituții de învățământ sau oricărui elev care are un cuvânt de spus și vede școala într-un mod modern.

# Structura proiectului



```
▼ Proiect-F4SS
  ▼ lib
    ▼ css
      /* style.css
    ▼ html
      <> footer.html
      <> intro.html
      <> modal.html
      <> navbar.html
    ▼ img
      1.jpg
    ▼ js
      /* app.js
      /* directives.js
      /* script.js
  ► node_modules
  ▼ view
    <> index.html
    /* export-collection.json
    /* index.js
    /* package-lock.json
    /* package.json
```

Structura folderului în care se afla proiectul este următoarea. În folderul principal se află fișierul `index.js`, împreună cu `package.json` și `package-lock.json` și încă trei foldere, unul pentru modulele node, al doilea conține

iar celălalt „lib” unde am pus fișiere în funcție de tipul lor, astfel în folderul „css” am `style.css` unde am adus mici modificari, în „html” sunt secțiunile paginii, în „img” este imaginea pe care am folosit-o în pagină (restul imaginilor folosite au fost preluate prin link de la alte pagini), iar în folderul „js” sunt scripturile.

# Prezentarea proiectului

Am vrut ca site-ul să fie de tip one page, deoarece scopul său este de a colecta feedback și a informa, iar acest lucru trebuie să se realizeze într-o manieră simplă și plăcută pentru utilizator.

Am început proiectul cu o mică inițializare (npm init) pentru a instala modulele Node, apoi am instalat librăriile de care avem nevoie pentru a rula local un server și pentru conectarea la baza de date (express și mongoose). Ulterior am adăugat body-parser pentru a prelua parametrii post, http pentru crearea unei instanțe de server la care se poate atașa Socket.io care este folosit să informeze toți utilizatorii conectați în cazul unei schimbări fără reload . Acestea au fost introduse în fișierul index.js din folderul principal.

Librăriile pe care le-am folosit sunt open source.

```
// includerea bibliotecilor pentru extinderea
// modulară a funcționalității engine-ului NodeJS
var express = require('express'),
    app = express(),
    server = require('http').createServer(app),
    io = require('socket.io')(server),
    mongoose = require('mongoose'),
    bodyParser = require('body-parser');
```

```
// inițializare middleware pentru preluare parametrii POST din request
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: true
}));
```

```
// acceptă clienți pe socket pentru informarea în momentul actualizării
io.on('connection', function (socket) {
  console.log('a user connected');
});

server.listen(3000, function () {
  console.log('App listening on port 3000');
});
```

Am continuat prin conectarea la baza de date MongoDB care rulează local, pentru dezvoltare am preferat să folosesc un server local, dar pentru implementare s-ar putea folosi un serviciu cloud.

Am declarat schema obiectelor pentru baza de date, pe baza căreia am făcut un model, din cauză că proiectul este în dezvoltare și nu am putut să implementez un sistem de autentificare pentru licee, datele le-am adăugat manual utilizând MongoDB Compass Community

```
// realizarea conexiunii la baza de date
mongoose.connect("mongodb://localhost/smartschools").then(
  () => {
    console.log('Connected to mongodb')
  },
  err => {
    console.log(err);
  }
);

// schema de baza pentru liceu
var schema = new mongoose.Schema({
  name: 'string',
  description: 'string',
  img: 'string',
  url: 'string',
  departments: 'array'
});

// model facut pe baza schemei
var school = mongoose.model('HighSchool', schema);
```

Pagina principală index.html din folderul „view” are următoarea structură:

```
<!DOCTYPE html>
<html ng-app="myApp">

<head>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/css/bootstrap.min.css"
  integrity="sha384-9gVQ4dYFwwWSjIDZnLEWnxCjeSWFphJiiGPXr1jddIh0Oegiu1Fw05qRgvFXOdJZ4" crossorigin="anonymous">
  <link rel="stylesheet" type="text/css" href="lib/css/style.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.0.10/css/all.css" integrity="
  sha384-d0P83n9kaQMcwJ8F4RJB66t2IwOKmrd46+porD/OvrJ+37WqIM7UoBtwH06Nlg" crossorigin="anonymous">
</head>

<body ng-controller="mainController" id="main">

  <navbar></navbar>

  <intro></intro>

  <footeer></footeer>

  <script src="/socket.io/socket.io.js"></script>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.25/angular.min.js"></script>
  <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.2.25/angular-route.js"></script>
  <script src="lib/js/app.js"></script>
  <script src="lib/js/directives.js"></script>
  <script src="lib/js/script.js"></script>
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.0/umd/popper.min.js" integrity="sha384-cs/
  chfZiN24E4KMATLdqdvszGxaGsi4hLG0zLXwp5UzB1LY//20VyM2taTB4QvJ" crossorigin="anonymous"></script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.0/js/bootstrap.min.js" integrity="
  sha384-uefMccjFJAIV6A+rW+L4AHf99KvxDjWSu1z9VI8SKNVmz4sk7buKt/6v9KI65qnm" crossorigin="anonymous"></script>
</body>

</html>
```

În header am inserat linkurile de la CDN-uri de la frameworkul Bootstrap, pe care l-am folosit, deoarece am vrut să ofer utilizatorului un site cu un design plăcut și responsive. CDN-ul de la FontAwesome l-am folosit, deoarece am avut nevoie de un icon. Este prezent și linkul de stilul modificat.

Pentru a putea prelua și afișa datele din baza de date am folosit AngularJS, iar în body sunt directivele paginii



Directivele sunt în folderul lib/js, în fișierul directives.js

```
//directivele paginii
app.directive('navbar', function () {
  return {
    restrict: 'E',
    templateUrl: '/lib/html/navbar.html'
  }
});

app.directive('intro', function () {
  return {
    restrict: 'E',
    templateUrl: '/lib/html/intro.html'
  }
});

app.directive('modal', function () {
  return {
    restrict: 'E',
    templateUrl: '/lib/html/modal.html'
  }
});

app.directive('footeer', function () {
  return {
    restrict: 'E',
    templateUrl: '/lib/html/footer.html'
  }
});
```

Pagina este secționată în mai multe secțiuni, datorită directiveilor.

Astfel meniul sau fișierul din lib/html/navbar.html are următoarea structură:

Din cauză că site-ul este de tip one-page nu am avut nevoie decât de o opțiune în meniu

```
<header>
  <nav class="navbar navbar-expand-md navbar-dark bg-dark">
    <a class="navbar-brand" href="#">Feedback for smart schools</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
      data-target="#navbarCollaparia" controls="navbarCollapse"
      aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarCollapse">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item active">
          <a class="nav-link" href="#">Acasă</a>
        </li>
      </ul>
    </div>
  </nav>
</header>
```

În intro.html din lib/html se află toată acțiunea!

Prima dată suntem întâmpinați de un jumbotron cu o imagine specifică, unde este prezentată tema paginii, încurajând utilizatorii să lase un feedback.



```
<main role="main">
  <!-- secțiunea de introducere cu un text informativ -->
  <section class="jumbotron text-center intro">
    <div class="box-light">
      <h1 class="display-4">Schimbarea începe cu o idee</h1>
      <p class="lead">Ai o sugestie, o reclamație sau doar dorești să ne împărtășești din experiența ta? <br> Ai venit în locul perfect! Alege liceul, apoi secțiunea unde s-ar încadra comentariul tău și fii cât mai sincer. Vrem să construim viitorul și nu putem face asta fără tine, avem nevoie permanent de feedbackul tău.</p>
    </div>
  </section>
```

Urmează secțiunea unde utilizatorii își pot spune părerea. Aici sunt generate automat cu ng-repeat carduri cu liceele, iar pentru a ușura „munca” utilizatorului în încercarea de a găsi liceul de interes, am adăugat un input pentru căutare care filtrează cardurile.

Cardurile au o imagine sugestivă, numele, o mică descriere, un link către pagina oficială a liceului și un button de tip dropdown unde apar departamentele pentru feedback. La alegerea unui departament se va deschide un modal unde utilizatorul va putea lăsa comentariul și va putea vedea celelate păreri ale utilizatorilor.

```

<!-- secțiunea unde sunt cardurile cu liceele -->
<div class="py-5 bg-light">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <form class="form-inline my-2 my-lg-0 padd">
          <!-- câmpul pentru căutarea liceului -->
          <input class="form-control mr-sm-2" type="search" placeholder="Caută liceul tău..." aria-label="Search" ng-model='searchText'>
        </form>
      </div>
    </div>
    <div class="row">
      <div class="card-columns">
        <!-- repetarea cardurilor în funcție de obiectele din baza de date filtrarea rezultatelor în funcție de căutare -->
        <div ng-repeat="school in data | filter:searchText">
          <div class="card mb-4 box-shadow">
            
            <div class="card-body">
              <h5 class="card-title">{{school.name}}</h5>
              <p class="card-text">{{school.description}} <a ng-href="{{school.url}}" ng-if="school.url" target="_blank">
                Pagina Liceului</a></p>
              <div class="dropdown">
                <!-- dropdown cu departamentele/secțiunile unde utilizatorii pot adăuga comentarii -->
                <button class="btn btn-secondary dropdown-toggle" type="button" id="dropdownMenu2" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Lasă-ne...</button>
                <div class="dropdown-menu" aria-labelledby="dropdownMenu2">
                  <!-- selectarea departamentului va deschide un modal pentru a lasa feedback -->
                  <button class="dropdown-item" type="button" ng-repeat="dept in school.departments" ng-click="open($parent.$index,$index)">{{dept.name}}</button>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</main>
<modal></modal>

```

Cardurile sunt generate automat în funcție de datele din MongoDB cu ajutorul AngularJS și a obiectelor adaugate manual in baza de date.

localhost:27017 STANDALONE MongoDB 3.6.4 Community

smartschools.highschools

	DOCUMENTS	TOTAL SIZE	AVG. SIZE	INDEXES	TOTAL SIZE	AVG. SIZE
	6	4.2KB	709B	1	36.0KB	36.0KB

Documents Explain Plan Indexes

FILTER { field: 'value' }

OPTIONS FIND

INSERT DOCUMENT VIEW LIST TABLE

Displaying documents 1 - 6 of 6

```

_id: ObjectId("5aea00f3e84d3704bc5deae3")
name: "Liceul Teoretic Neagoe Basarab Oltenița"
> departments: Array
description: "In toata istoria sa, acest liceu, prin rezultatele activitatii instruc..."
img: "https://geaninalisandru.files.wordpress.com/2009/05/pic_2537.jpg?w=300"
url: "https://www.nbasarab.ro"
__v: 0

_id: ObjectId("5aea0114e84d3704bc5deae4")
name: "Colegiului Național de Informatică Piatra-Neamț"
> departments: Array
description: "Ne imaginăm că vom reuși să transformăm procesul de învățare într-o no..."
img: "http://cni.nt.edu.ro/imagini_liceu/efect/0.jpg"
url: "http://cni.nt.edu.ro/new/"
__v: 0

```





# Modalul se deschide datorită funcției din fișierul localizat în lib/js/script.js

```
//functia pentru deschiderea modalului
function openModal() {
    $('#deptModal').modal('show');
}
```

```
<!-- Modalul afisat pentru a lasa feedback la departamentul selectat -->
<div class="modal fade" id="deptModal" tabindex="-1" role="dialog" aria-labelledby="deptModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered modal-lg" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <!-- titlul modalului -->
        <h5 class="modal-title" id="deptModalLongTitle">{{data[selectededschool].departments[selecteddept].name}} - {{data[selectededschool].name}}</h5>
        <!-- butonul de inchidere al modalului -->
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        <div class="container">
          <div class="row">
            <!-- formularul (cu apel ajax) pentru feedback -->
            <div class="col-md-6">
              <div class="form-group">
                <input type="text" class="form-control" placeholder="Nume" id="getUserNume">
              </div>
            </div>
            <div class="col-md-6">
              <div class="form-group">
                <select class="form-control" id="id-element-select">
                  <option>Părinte</option>
                  <option>Elev</option>
                  <option>Fost elev</option>
                  <option>Profesor</option>
                  <option>Bunic</option>
                </select>
              </div>
            </div>
            <div class="col-md-12">
              <div class="form-group">
                <textarea class="form-control" rows="3" id="getContent" placeholder="Lasă-ne mesajul tau aici!"></textarea>
              </div>
              <div class="padd">
                <button class="btn btn-outline-primary float-right" ng-click="postreview(selecteddept)">Adaugă</button>
              </div>
            </div>
          </div>
        </div>
      </div>
      <div class="modal-body">
        <div class="container">
          <div class="row">
            <!-- secțiunea din modal unde apar comentariile -->
            <div class="col-md-12" ng-repeat="review in data[selectededschool].departments[selecteddept].reviews">
              <div class="card mb-3 w-100">
                <div class="card-header">
                  <strong>{{review.name}} - {{review.type}}</strong>
                  <span class="text-muted">{{review.time | date : 'short'}}</span>
                </div>
                <div class="card-body">
                  <p class="card-text">{{review.content}}</p>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

Pentru a face legatura modalului cu obiectul din baza de date, in fisierul app.js din folderul lib/js, am creat o funcție care memorează variabilele pentru afișarea recenziilor corespunzătoare în modal, apoi îl deschide. Pentru funcția de adaugare am făcut o altă functie care ia valorile din campuri si le trimite la server printr-o cerere post.

```
var app = angular.module("myApp", []);
var socket = io();

// apelează funcția pentru actualizarea paginii
socket.on('update', function (msg) {
    angular.element(document.getElementById('main')).scope().get();
});

// request pentru adăugare
function postReview(schoolid, dept) {
    var sel = document.getElementById("id-element-select");
    var strUser = sel.options[sel.selectedIndex].text;
    $.post('/addReview', {
        "id": schoolid,
        "dept": dept,
        "content": document.getElementById('getcontent').value,
        "userName": document.getElementById('getUserName').value,
        "userType": strUser
    }, function (response) {
        console.log(response)
    });
    document.getElementById('getcontent').value = ''; // curăță caseta de comentarii
    document.getElementById('getUserName').value = ''; // curăță caseta cu numele
}

app.controller('mainController', function ($scope, $http, $window) {
    // memorează variabilele pentru afișarea recenziilor corespunzătoare în modal, apoi îl deschide
    $scope.open = function (school, dept) {
        $scope.selectedschool = school;
        $scope.selecteddept = dept;
        $window.openModal();
    }
    $scope.postreview = function (index) {
        $window.postReview($scope.data[$scope.selectedschool]._id, index);
    }
    // încarcă datele de pe API în scope
    $scope.get = function () {
        $http.get("/list")
            .then(function (response) {
                $scope.data = response.data;
            });
    }
    // încărcăm o dată la început
    $scope.get();
});
```

Funcția din index.js pentru adăugarea unui review, prima dată se creează obiectul, apoi se adaugă în arrayul obiectului, iar în cazul unei erori se afișează un mesaj

```
// adaugarea review
app.post('/addReview', function (req, res) {
  var schoolid = req.body.id,
      dept = req.body.dept,
      name = req.body.userName,
      type = req.body.userType,
      text = req.body.content,
      build = "departments." + dept + ".reviews"; // pentru crearea etichetei json
  console.log(dept);
  school.update({ // filtrează școli după ID
    "_id": schoolid
  }, { // adaugă în array
    "$push": {
      [build]: // nume variabil json, creat după indexul respectivului departament
      {
        "name": name,
        "type": type,
        "content": text,
        "time": Date.now()
      }
    }
  },
  function (err, raw) {
    if (err) {
      res.send(err); // răspunsul în cazul unei erori
      console.log('Error while adding')
    }
    res.send(raw); // răspuns
    io.emit('update', {
      for: 'everyone'
    }); // emite informare actualizare
    console.log('Added review');
  }
});
});
```

În finalul paginii se află footerul unde liceele interesate, care doresc să apară pe platformă, pot găsi adresa de contact.

© 2018 · Titrat Cristina

Până la finalizarea platformei, vă rugăm să ne contactați, pentru adăugare manuală a liceului, la adresa:

 titratcristina@gmail.com

```
<footer class="container">
  <div class="row">
    <p class="col-md-4">&copy; 2018 &middot; Empowersoft &middot; Titrat Cristina</p>
    <p class="col-md-8">Până la finalizarea platformei, vă rugăm să ne contactați, pentru adăugare manuală a liceului, la adresa:<br>
      <i class="far fa-address-card"></i> titratcristina@gmail.com</p>
  </div>
</footer>
```

Iar la sfârșitul fișierului index.html se află scripturile de la AngularJS, socket.io, app.js, script.js, index.js, directives.js și bootstrap.js



# Indicații de instalare a proiectului

Pentru instalarea proiectului, se extrage folderul „Proiect-F4SS” din arhivă.

Pe calculator trebuie să existe instalate baza de date MongoDB cu MongoDB Compass Community și Node.js.

Din cmd, in folderul proiectului, se introduce `npm install`, pentru instalarea librariilor și a modulelor node. Apoi din folderul MongoDB se executa comanda „`mongod`” pentru pornire, apoi in folderul proiectului pentru pornirea serverului se executa `node index.js`. Pentru afișarea liceelor se va importa in MongoDB Compass Community fișierul json export-collection, după crearea unei baze de date cu numele „smartschools” si o colectie „highschools”

```
C:\Users\Cristina PC\Desktop\Proiect-F4SS>node index.js
App listening on port 3000
Connected to mongodb
a user connected
a user connected
a user connected
a user connected
a user connected
a user connected
a user connected
1
Added review
```

```

C:\Program Files\MongoDB\Server\3.6\bin>mongo
2018-05-03T13:40:17.252-0700 I CONTROL [initandlisten] MongoDB starting : pid=1920 port=27017 dbpath=C:\data\db\ 64-bit host=DESKTOP-38H61H3
2018-05-03T13:40:17.252-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2018-05-03T13:40:17.253-0700 I CONTROL [initandlisten] db version v3.6.4
2018-05-03T13:40:17.254-0700 I CONTROL [initandlisten] git version: d0181a711f7e7f39e60b5aeb1dc7097bf6ae5856
2018-05-03T13:40:17.254-0700 I CONTROL [initandlisten] OpenSSL version: OpenSSL 1.0.2o-fips 27 Mar 2018
2018-05-03T13:40:17.254-0700 I CONTROL [initandlisten] allocator: tcmalloc
2018-05-03T13:40:17.255-0700 I CONTROL [initandlisten] modules: none
2018-05-03T13:40:17.257-0700 I CONTROL [initandlisten] build environment:
2018-05-03T13:40:17.258-0700 I CONTROL [initandlisten]   distmod: 2008plus-ssl
2018-05-03T13:40:17.260-0700 I CONTROL [initandlisten]   distarch: x86_64
2018-05-03T13:40:17.261-0700 I CONTROL [initandlisten]   target_arch: x86_64
2018-05-03T13:40:17.263-0700 I CONTROL [initandlisten] options: {}
2018-05-03T13:40:17.281-0700 I - [initandlisten] Detected data files in C:\data\db\ created by the 'wiredTiger' storage engine, so setting the active storage eng
ine to 'wiredTiger'.
2018-05-03T13:40:17.282-0700 I STORAGE [initandlisten] wiredtiger_open config: create,cache_size=1491M,session_max=20000,eviction=(threads_min=4,threads_max=4),config_
base=false,statistics=(fast),cache_cursors=false,log=(enabled=true,archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),statistics_log=(wa
it=0),verbose=(recovery_progress),
2018-05-03T13:40:18.274-0700 I STORAGE [initandlisten] wiredtiger message [1525380018:273724][1920:140717780722000], txn-recover: Main recovery loop: starting at 10/41
472
2018-05-03T13:40:18.975-0700 I STORAGE [initandlisten] wiredtiger message [1525380018:975223][1920:140717780722000], txn-recover: Recovering log 10 through 11
2018-05-03T13:40:19.826-0700 I STORAGE [initandlisten] wiredtiger message [1525380019:355491][1920:140717780722000], txn-recover: Recovering log 11 through 11
2018-05-03T13:40:20.087-0700 I CONTROL [initandlisten] wiredtiger message [1525380019:826004][1920:140717780722000], txn-recover: Set global recovery timestamp: 0
2018-05-03T13:40:20.091-0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-05-03T13:40:20.094-0700 I CONTROL [initandlisten] Read and write access to data and configuration is unrestricted.
2018-05-03T13:40:20.096-0700 I CONTROL [initandlisten] ** WARNING: This server is bound to localhost.
2018-05-03T13:40:20.097-0700 I CONTROL [initandlisten] Remote systems will be unable to connect to this server.
2018-05-03T13:40:20.102-0700 I CONTROL [initandlisten] Start the server with --bind_ip <address> to specify which IP
2018-05-03T13:40:20.104-0700 I CONTROL [initandlisten] addresses it should serve responses from, or with --bind_ip_all to
2018-05-03T13:40:20.107-0700 I CONTROL [initandlisten] bind to all interfaces. If this behavior is desired, start the
2018-05-03T13:40:20.108-0700 I CONTROL [initandlisten] server with --bind_ip 127.0.0.1 to disable this warning.
2018-05-03T13:40:20.110-0700 I CONTROL [initandlisten]
2018-05-03T13:40:20.115-0700 I CONTROL [initandlisten] ** WARNING: The file system cache of this machine is configured to be greater than 40% of the total memory. This
2018-05-03T13:40:20.117-0700 I CONTROL [initandlisten] can lead to increased memory pressure and poor performance.
2018-05-03T13:40:20.120-0700 I CONTROL [initandlisten] See http://dochub.mongodb.org/core/wt-windows-system-file-cache
2018-05-03T13:40:20.121-0700 I CONTROL [initandlisten]
2018-05-03T13:40:21.039-0300 I FTDC [initandlisten] Initializing full-time diagnostic data capture with directory 'C:\data\db\diagnostic.data'
2018-05-03T13:40:21.049-0300 I NETWORK [initandlisten] waiting for connections on port 27017
2018-05-03T13:40:22.193-0300 I NETWORK [listener] connection accepted from 127.0.0.1:51678 #1 (1 connection now open)
2018-05-03T13:40:25.090-0300 I NETWORK [conn1] end connection 127.0.0.1:51678 (0 connections now open)
2018-05-03T13:40:43.155-0300 I NETWORK [listener] connection accepted from 127.0.0.1:51706 #2 (1 connection now open)
2018-05-03T13:40:43.180-0300 I NETWORK [conn2] received client metadata from 127.0.0.1:51706 conn2: { driver: { name: "nodejs", version: "3.0.7" }, os: { type: "window
s_NT", name: "win32", architecture: "x64", version: "10.0.16299" }, platform: "Node.js v8.9.1, LE, mongodb-core: 3.0.7" }

```

## Create Database

### Database Name

### Collection Name

☐ Capped Collection ⓘ

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)

CANCEL

CREATE DATABASE

# Bibliografie

- Descrieriile liceelor sunt preluate de pe siteurile originale
- Imaginile sunt introduse prin linkuri de la alte pagini