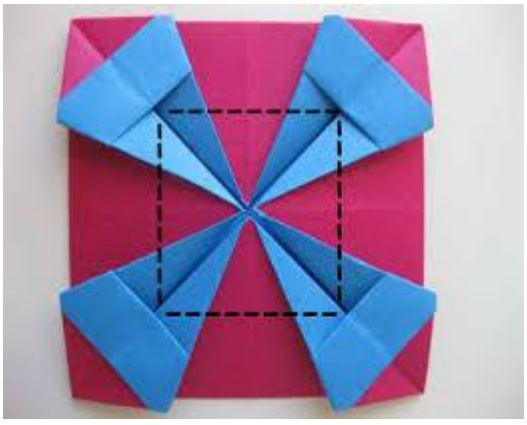
Portal Origami

Pagini Web



Corcodel Iulia-Maria
Gavrea Andrei

Cuprins

- Tehnologii folosite
- Tema proiectului
- Prezentarea proiectului

Tehnologii folosite

Backend:

1. NodeJs
2. Express
3. Multer
4. Bcrypt
5. Mongoose(pentru baza de date MongoDB)

Fontend

1. HTML
2. CSS
3. AngularJs

Editoare folosite au fost:

1. Sublime 3
2. Dreamweaver 2015
3. Nano

Tema proiectului

Am ales ca tema a proiectului nostru “arta japoneza-origami”, astfel am creat un site care sa uneasca oamenii pasionati de aceasta, in special pe cei din Romania deoarece in tara noastra aceasta arta nu este indeajuns de promovata. Utilizatorii isi pot expune creatiile lor sau pot promova modele din intreaga lume. De asemenea, acestia pot reactiona pozitiv sau negativ la lucrarile celorlalti utilizatori si de a adauga comentarii prin care sa isi exprime opinia in legatura cu ele. Membrii comunitatii isi pot personaliza profilul prin adaugarea unei imagini de profil.

Prezentarea proiectului

Proiectul a fost gandit ca fiind un web-application, care functioneze si sa furnizeze datele utilizatorilor printr-un RESTful API.

Fisierul de pornire este “app.js”, unde am pus modulele, am definit portul si fisierele statice, iar la final am adaugat socket.io pentru actualizarea in real time a aprecierilor.

```
var express= require("express");
var bodyParser= require("body-parser");
const mongoose = require('mongoose');
var app= express();
const  port = parseInt(process.env.PORT, 10) || 8080;
var db= require("./models/connection.js");
var session = require('express-session');
var MongoStore = require('connect-mongo')(session);
app.set('port', port);
const server = app.listen(app.get('port'), ()=>{
    console.log(' opened the gates on port ' + app.get('port'));
});
var io = require('socket.io')(server);
var jsonParser = bodyParser.json();
var urlencodedParser = bodyParser.urlencoded({ extended: true });
app.use(jsonParser);
app.use(urlencodedParser);
app.use(session({
  secret: 'ioolia',
  resave: [redacted]
  saveUninitialized: false,
  rolling:true,
 cookie: { httpOnly: false  ,  maxAge: 600000},
  store: new MongoStore({
    mongooseConnection: db
  })
}));
app.use("/view", express.static(__dirname + '/view'));

app.use("/origami", express.static(__dirname + '/origami'));
app.use("/uploads", express.static(__dirname + '/uploads'));
require('./routes')(app);

app.get('*', (req, res) => {
    res.sendFile(__dirname + '/view/index.html');
console.log(req.session.username);
console.log(req.session.cookie);
});

io.on('connection', function(socket){
  console.log('a user connected '+ socket.id);

socket.on('addlike',  (data) => {
io.emit('addlike',data);
});
socket.on('adddislike',  (data) => {
io.emit('adddislike',data);
});
socket.on('addundislike',  (data) => {
io.emit('addundislike',data);
});
socket.on('addunlike', (data) =>{
console.log('server side o yeee '+ socket.id);
```

Ulterior, am facut conexiunea si am inceput sa creez schemele si modelele pentru baza de date.

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/test2');
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function() {
console.log("gata");
});
module.exports = db;
```

```
var mongoose = require('mongoose');

const OrigamiSchema = new mongoose.Schema({
    userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
    origamititle: {
        type: String,
        unique: true,
    required: true,
    },
    origamidescription: {
        type: String,
    },
origamipath:{type:String,require:true},
    commentaArray: [
        {
            userId: { type: mongoose.Schema.Types.ObjectId, ref: 'User' },
        title: {
                type: String,
            },

            comment: {
                type: String,
            },
            Date: {
                type: Date,  default: Date.now(),
            },
        }
    ],
    tags: [{type: String}],
    likes: {type: Array, default: []},
    dislikes: {type: Array, default: []},
    Date: { type: Date, default: Date.now },
});

module.exports = mongoose.model('Origami', OrigamiSchema);
```

Pe langa schema contului de utilizator, am introdus o functie care hashuieste parola utilizatorului pentru a o salva criptata in baza de date, contra atacurilor, si am creat un “statics” pentru autentificarea utilizatorilor care cauta numele de utilizator sau emailul si daca ia parola introdusa si o compara cu cea din baza de date.

```
  },
  birthdate: {
    type: Date
  },
  gender:{
    type: Boolean //0 for boys 1 for girls
  },
  registerdata:{
    type:Date,
    default: Date.now()

  },
  profilepath:{type: String},
  admin: { type: Boolean, default: false },
  liked: {type: Array, default: []},
});
UserSchema.statics.authenticate = function (email, password, callback) {
  User.findOne({$or: [
    {email: email},
    {username: email}
]})
    .exec(function (err, user) {
      if (err) {
        return callback(err)
      } else if (!user) {
        var err = new Error('User not found.');
        err.status = 401;
        return callback(err);
      }
      bcrypt.compare(password, user.password, function (err, result) {
        if (result === true) {
          return callback(null, user);
        } else {
          return callback();
        }
      })
    });
}
UserSchema.pre('save', function (next) {
  var user = this;
  bcrypt.hash(user.password, 10, function (err, hash){
    if (err) {
      return next(err);
    }
    user.password = hash;
    next();
  })
});
var User = mongoose.model('User', UserSchema);
module.exports = User;
```

Dupa crearea modelelor, am creat rutele api-ului serverului. In folderul *routes,* fisierul index.js reprezinta fisierul care face legatura intre api si aplicatie.

```
const express = require('express');
const router = express.Router();
const user = require('./userRoutes.js');
const origami = require('./origamiRoutes.js');


module.exports = (app) => {

router.use('/', origami);
router.use('/', user);
router.route('/test')
    .get((req,res) => {
res.status(200).json({ message: 'Connected!' });

})
    .post( (req,res) => {
res.status(200).json({ message: 'Connected!2' });
});
app.use('/api', router);

}
```

```
.post(upload.single('file'),function(req, res, next) {
        if (req.session.username == "undefined") {

            var err = new Error('No user atm.');
        err.message="No user atm.";
            err.status = 401;
            return next(err);

        } else {
                    console.log(req.body);
            if (req.body.origamidescription &&
                req.body.origamititle && req.file ) {
                console.log(req.body);
                console.log(req.file);
                var bitmap = fs.readFileSync('origami/' + req.file.filename).toString('hex', 0, 4)
                console.log(bitmap);
                if (!checkMagicNumbers(bitmap)) {
                    fs.unlinkSync('origami/' + req.file.filename);
                    var err = new Error('Not a picture');
                    err.status = 401;
                    return next(err);

                } else {
                    var newUpload = {
                        userId: req.session.userid,
                        origamipath: 'origami/' + req.file.filename,
                        origamititle:req.body.origamititle,
                        origamidescription:req.body.origamidescription,
                        tags: req.body.tags,

};




                    Origami.create(newUpload, function(error, user) {
                        if (error) {
                            return next(error);
                        } else {
```

Atunci cand utilizatorul trimite o poza, fie de profil fie pentru origami, “numerele magice” ale ei sunt verificate pentru a verifica veridicitatea pozei, aceste parti fiind backendul aplicatiei.

exemplu in poza din pagina de mai sus

```
router.route('/register')

        .post( function (req, res, next){
if (req.body.email &&
   req.body.username &&
    req.body.password &&
    req.body.name &&
    req.body.surname &&
    req.body.gender &&
  req.body.birthdate
) {

    var userData = {
      email: req.body.email,
      username: req.body.username,
      password: req.body.password,
      name:req.body.name,
      surname: req.body.surname,
      birthdate: req.body.birthdate,
      gender: req.body.gender
}

    User.create(userData, function (error, user) {
      if (error) {
        error.message="Numele de utilizator/parola/emailul au fost folosite deja";
        return res.status(500).send(error);
      } else {
  req.session.username = user.username;
      req.session.userid=user._id;
 res.success=1;
res.data={};

        res.data.username=user.username;
return res.send(user.username);


 }
    });

  }  else {
    var err = new Error('Toate campurile sunt obligatorii.');
    err.status = 400;
    err.message="Toate campurile sunt obligatorii.";
     res.status(401).send('Toate campurile sunt obligatorii.');
  }
});
```

Apoi, am trecut la crearea frontendului, acesta fiind realizat in AngularJS.

```
angular.module('ori', ['ui.router', 'ui.bootstrap', 'ngMaterial','ngCookies',"ngMessages",'ngFileUpload']);
```

Am definit rutele aplicatiei, am stabilit templateul, url-ul si contollerul fiecarei pagini.

```
angular.module('ori')
  .config(function($stateProvider, $urlRouterProvider) {
    $urlRouterProvider.otherwise('/contacts');
    $stateProvider.state('contacts', {
        url: '/contacts',
        templateUrl: '/view/partial1.html',
        controller: 'origamiController'
      })
      .state('account', {
        url: '/account',
        abstract: true,
        templateUrl: '/view/parent.html',
        controller: function($scope) {
          $scope.title = 'poza';
        }
      })
      .state('account.login', {
        url: '/login',
        template: '<log></log>',
        controller:  'userController'


      })
      .state('account.register', {
        url: '/register',
        template: '<register></register>',
        controller:  'userController'

      })
      .state('acasa', {
        url: '/acasa',
        templateUrl: '/view/acasa.html',
        controller: 'origamiController'
      })
  .state('origami', {
        url: '/origami/:origami',
        templateUrl: '/view/origami.html',
        controller: 'origamiController'
      })

      .state('profile', {
        url: '/profile/:username',
        templateUrl: '/view/profile.html',
        controller:  'userController'
      });
  })
```

Am utilizat directivele pentru usurarea crearii templateului, urmand in viitor sa le include diferite functii in acestea.

```
angular.module('ori')
  .directive('navbar', function() {
    return {
      restrict: 'E',
      templateUrl: '/view/navbar.html'
    };
  })
  .directive('log', function() {
    return {
      restrict: 'E',
      templateUrl: '/view/log.html'
    };
  })
.directive('profile', function() {
  return {
    restrict: 'E',
    templateUrl: '/view/profile.html'
  };
})
  .directive('register', function() {
    return {
      restrict: 'E',
      templateUrl: '/view/forminreg.html'
    };
  })
```

```
<div class="container-fluid" style="padding: 0">
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark" >
  <a class="navbar-brand" id="site"   ui-sref='contacts'>Origami</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent"
aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarSupportedContent">
    <ul class="navbar-nav mr-auto">
      <li class="nav-item active">
        <a class="nav-link" ui-sref='acasa'>Acasa </a>
      </li>
      <li class="nav-item" ng-hide="username">
        <a class="nav-link" ui-sref='account.login'>Inregistrare/Autentificare</a>
      </li>
       <li class="nav-item" ng-show="username">
        <a class="nav-link" ui-sref='profile({username: username})'>Profil</a>
      </li>
            <li class="nav-item" ng-show="username">
        <a id="delog" class="nav-link" ng-click="deletelogin()" >Delogare</a>
      </li>

    </ul>

  </div> </nav>
</div>
```

Paginile cu origami si profilul au fost create prin cereri la apiul de pe server si au fost afisate intr-un for. Respectiv, daca poza de profil este inexistenta se va pune o poza de profil de default .

```
<navbar></navbar>
<div class="row">
<div class="col-2">

            </div>

<div  class="container text-center fundal col-6">
  <p id="titlu">{{profile.username}}</p> <img class="img-fluid mx-auto" ng-show="profile.profilepath "  ng-src="/{{profile.profilepath}}"  style="width:15em ;margin-top:1em;"/>
<img class="img-fluid mx-auto" ng-src="/view/photos/defaultavatar.png" ng-hide="profile.profilepath "   style="width:15em ;margin-top:1em;"/>
 <div class="info">
  <p>Nume:{{profile.name}}</p> <p>Prenume: {{profile.surname}}</p> <p ng-show="profile.gender" >Sex:Feminin </p>
 <p ng-hide="profile.gender" >Sex:Masculin </p>
 <p>Si-a creat
contul pe: {{profile.registerdata | date}} </p> <p> Data nasterii: {{profile.birthdate | date}}</p> </div> </div></div>
<div class="row">
<div class="col-2">

            </div>
<div class="container text-center " id="poza">
<button type="button" ng-show="profile.username==username "  class="btn btn-primary text-center buton " data-toggle="modal" data-target="#Profil">
  Adauga o noua poza de profil </button></div></div>
 <!-- Modal -->
 <div class="modal fade" id="Profil" tabindex="-1" role="dialog" aria-labelledby="exampleModalLongTitle" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLongTitle">Poza de profil</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
        <form ng-submit="submit()">
      <div class="modal-body">



    <legend>Pune-ti poza de profil aici:</legend>
    <input type="file" class="form-control-file" name="file" id="file" ng-model="upload.file"
ngf-select ngf-max-size="25MB" required/>
    <br/>




      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Inchide</button>
        <button type="submit" class="btn btn-primary">Salveaza</button>
      </div> </form>
    </div>
  </div>
</div>
```

Am utilizat *service-uri* pentru a da cererile la server.

```
angular.module("ori").factory('userService', ["$http", function($http) {

  return {
authenticate:function(data){
configObject={
      method: 'POST',
      url: '/api/register',
      data: data
     }
        return $http(configObject);
},
 login:function(data){
configObject={
      method: 'POST',
      url: '/api/login',
      data: data
     }
      return $http(configObject);
 },
getUsername:function(){
configObject={
      method: 'GET',
      url: '/api/login'
     }
      return $http(configObject);
 },

getProfile:function(data){
configObject={
      method: 'GET',
      url: '/api/users/'+data
}
      return $http(configObject);
 },
deleteUser:function(){
configObject={
      method: 'DELETE',
      url: '/api/login'
     }
      return $http(configObject);
 }


} }]);
```

In fiecare controller se afla functiile fie pentru partea de origami fie pentru partea de utilizator impreuna cu variabilele aferente, respectiv la controllerul origami am folosit socket-uri pentru actualizarea aprecierilor in timp real catre toti utilizatorii activi.

```
    if(res.data=="4"){
   socketService.emit('adddislike',{data: data,user:$rootScope.username});
                console.log( "aici este DOAR dislike");
  console.log(res);
 }
   else if(res.data=="5"){
socketService.emit('adddislike',{data: data,user:$rootScope.username});
                console.log( "aici este dislike");
  console.log(res);

 socketService.emit('addunlike',{data: data,user:$rootScope.username});
                console.log( "aici este unlike");
  console.log(res);
 }



  else{ console.log("aici e undislike");
socketService.emit('addundislike', {data: data,user:$rootScope.username});
     for(var i=0;i<$scope.origami.length;i++){
    if($scope.origami[i]._id==data){
      console.log("am gasti elementul "+i);

}
}

}
},

        function errorCallback(res) {
            console.log('Error: ' + res.data);
     $state.go('account.login');
        });


}
socketService.on('addlike', function(data) {
console.log(data);
  for(var i=0;i<$scope.origami.length;i++){
                if($scope.origami[i]._id==data.data){
                        console.log("am gasti elementul "+i);
 $scope.origami[i].likes.push(data.username);
 $scope.origg.likes++;
$scope.$applyAsync();
        console.log("Buna lumeee");
    }
}
```

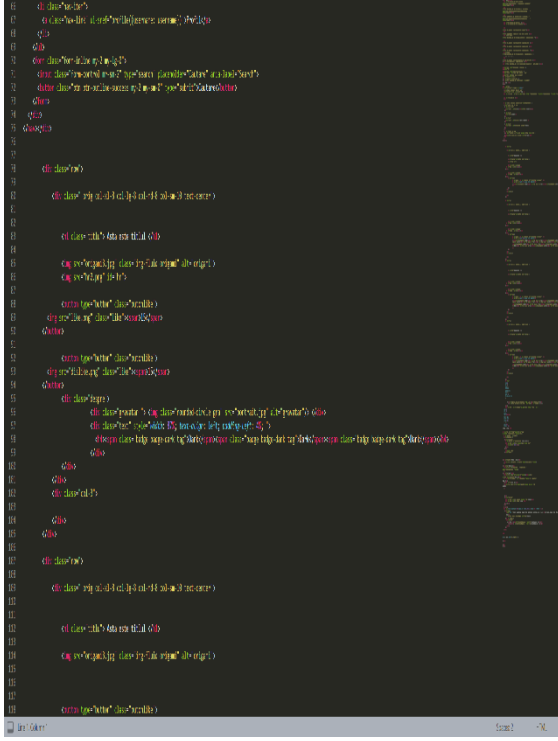
```
$scope.uploadTag = function(){
  $scope.upload.tags.push($scope.tag);
}
$scope.submit = function(){
console.log($scope);
 Upload.upload({
        url: '/api/origami',
        method: 'post',
        data: $scope.upload,
      }).then(function (response) {
  console.log("yeaaa boyyyyyyy");
        console.log(response.data);
  $("#origami").modal('hide');
  $scope.tag="";
        $scope.upload = {};
      })


}
$scope.sendcomment = function(){
origamiService.postCommentOrigami($stateParams.origami,$scope.upld)
                  .then(function successCallback(res) {
              $scope.origg = res.data;
              console.log($scope.origg);
              console.log(res);
        },
        function errorCallback(res) {
              console.log('Error: ' + res.data);
   $state.go('account.login');
        });
}
 $scope.removeItem = function(index){
    $scope.upload.tags.splice(index, 1);
  }
```

```
<navbar></navbar>

 <div class="row" ng-repeat="orig in origami">
   <div class=" orig col-xl-8 col-lg-8 col-md-8 col-sm-10 text-center">
       <a class="gra" ui-sref='origami({origami:orig._id })' >  <h1 class="titlu">   {{orig.origamititle}} </h1></a>
      <img ng-src="/{{orig.origamipath}}"   class="img-fluid origami"/>


<img src="/view/icons/hr2.png" id="hr">
 <button type="button" class="butonlike" id="likes{{orig._id}}"  ng-click="like(orig._id)">
                                <img ng-src="/view/icons/like.png" class="like"><span>{{orig.likes.length}}</span>
                            </button>

              <button type="button" class="butonlike"  ng-click="dislike(orig._id)">
                                <img ng-src="/view/icons/dislike.png" class="like"><span>{{orig.dislikes.length}}</span>
                            </button>

<div class="despre">


              <div class="gravatar">    <a class="gra" ui-sref='profile({username:orig.userId.username })'>
 <img class=" img-fluid rounded-circle gra" src="/{{orig.userId.profilepath}}" ng-show="orig.userId.profilepath" />
 <img class=" img-fluid rounded-circle gra" ng-hide="orig.userId.profilepath"  src="/view/photos/defaultavatar.png" />

</a>  </div>

              <div class="text"  style="width: 87%; text-align: left; padding-left: 4%;">
                 <h4>
                 <span  ng-repeat="tag in orig.tags " class="badge badge-dark tag">{{tag}}</span></h4>
              </div>


     </div>
   </div>
</div>
<button type="button" ng-show="username"  class="btn btn-primary" data-toggle="modal" data-target="#origami"  style="
    right: 0;
    left:  0;
    bottom: 0;
    right: 0!important;
    float:  right;
    position: sticky;
    z-index: 3000;
    /* top: 0; */
"> Adauga un origami</button>
```

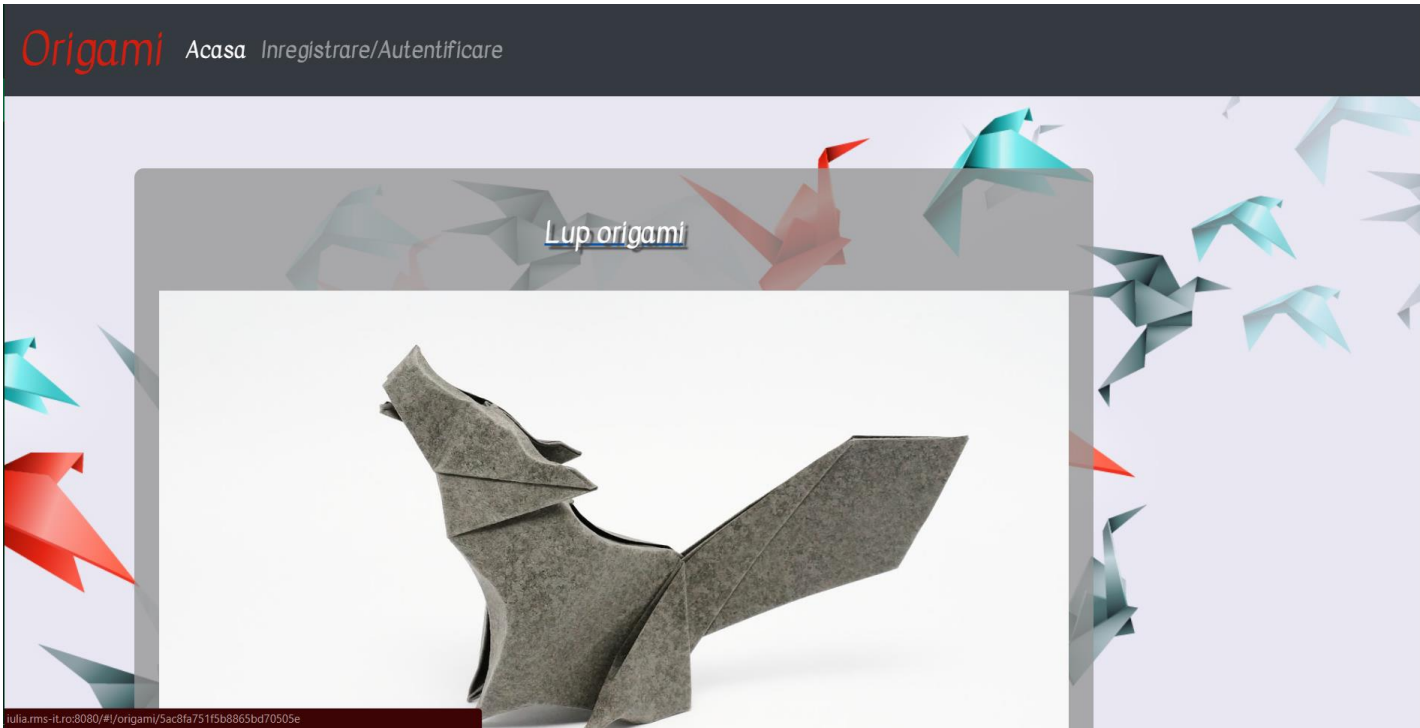
Pe partea de design am ales sa folosim Bootstrap 4, CSS3 si HTML5. Meniul se modifica in functie de latimea ecranului, la monitoare avand linkuri catre toate functionalitatile paginii, iar la telefoane fiind vizibil doar titlul Origami, urmand ca celelalte linkuri sa apara in urma unui dropdown.

Pagina de start are rolul de a expune scopul site-ului si te indeamna sa iti faci cont sau sa te loghezi pentru a adauga origami sau pentru a le aprecia/comenta pe cele existente.

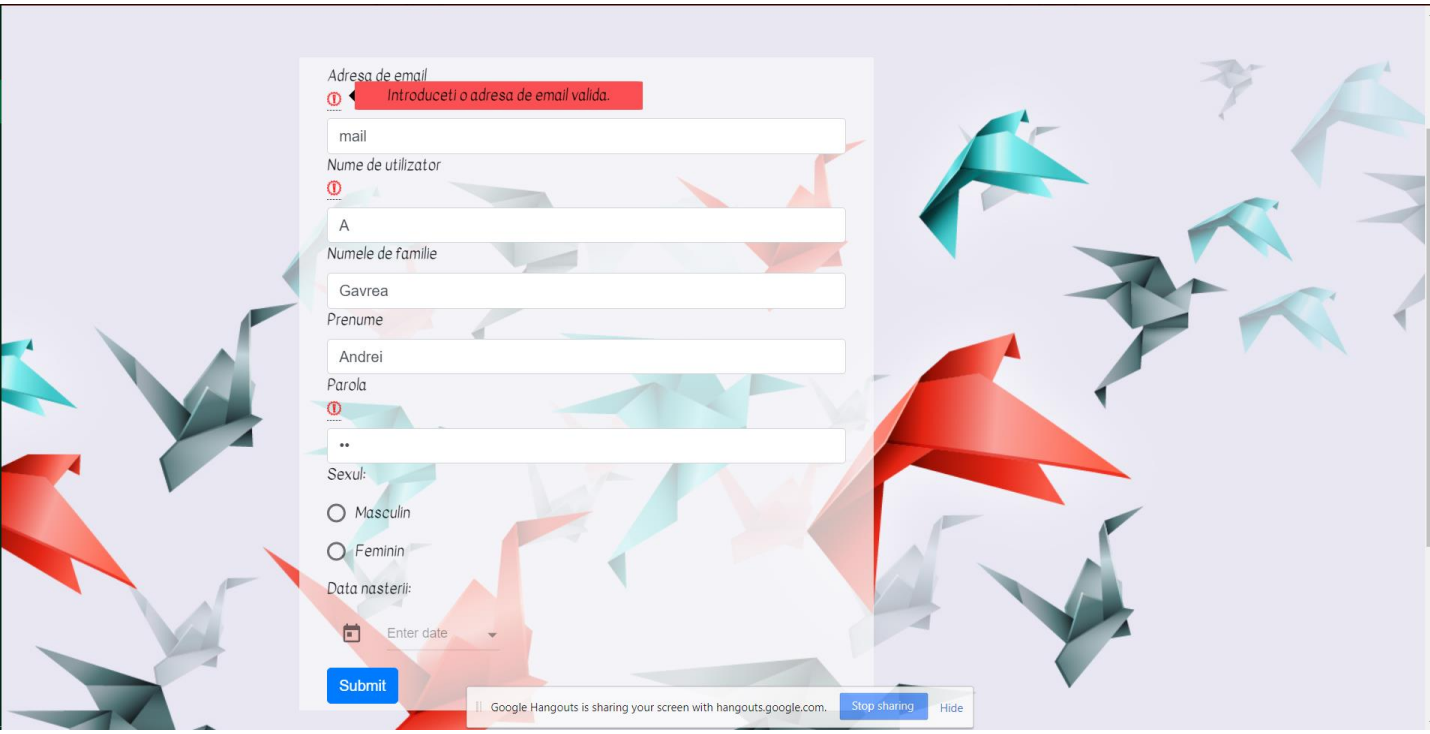
Pentru articole am ales un design simplu, acestea fiind alcatuite dintr-un titlu, o imagine reprezentativa a modelului origami si o descriere a sa urmata de datele autorului postarii si de numarul de likeuri/dislikeuri aferente.



Footerul cuprinde informatiile atat informatiile despre realizatorii site-ului, cat si linkul catre github



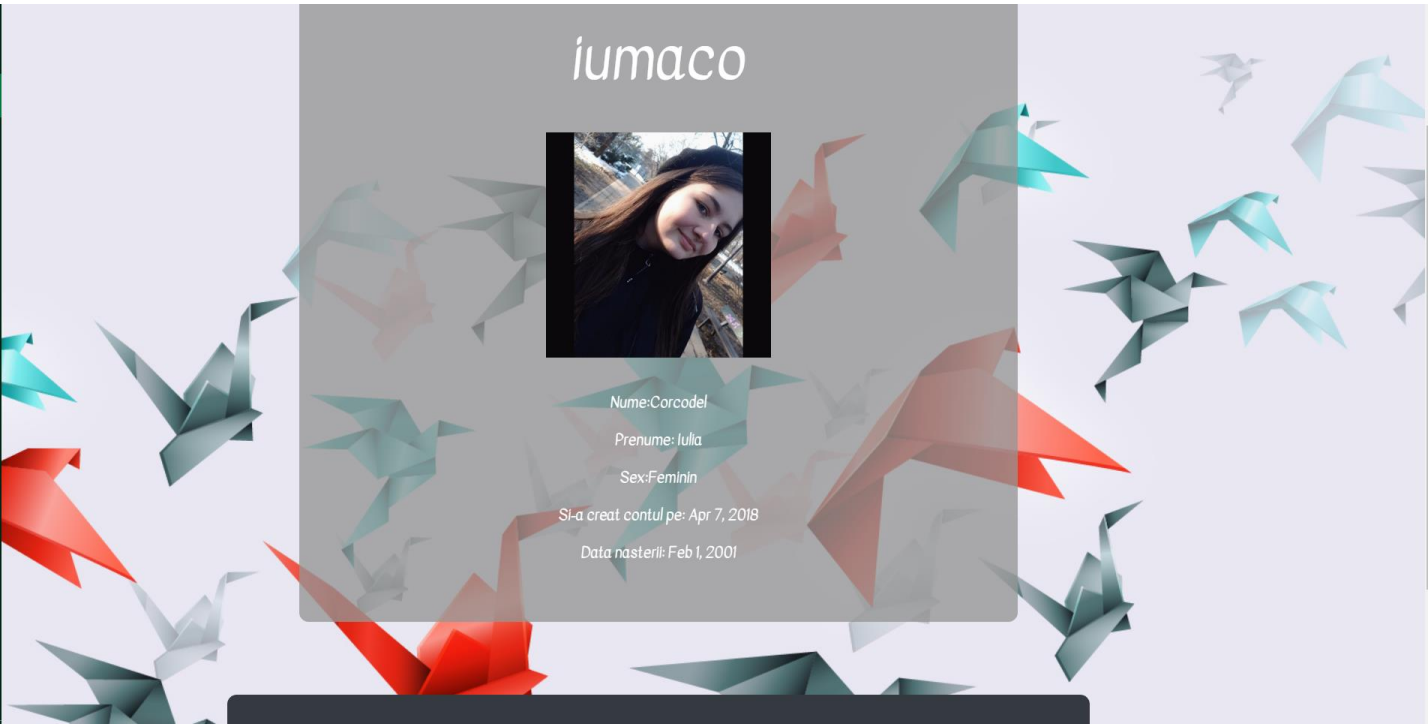
Daca te-ai decis sa iti faci un cont pe site-ul nostru vei fi directionat pe pagina de inregistrare, unde vei complete mai departe formularul aferent. Daca datele introduse nu respecta cerintele campului, vei fi semnalat pe parcursul inregistrarii, dar si la final.

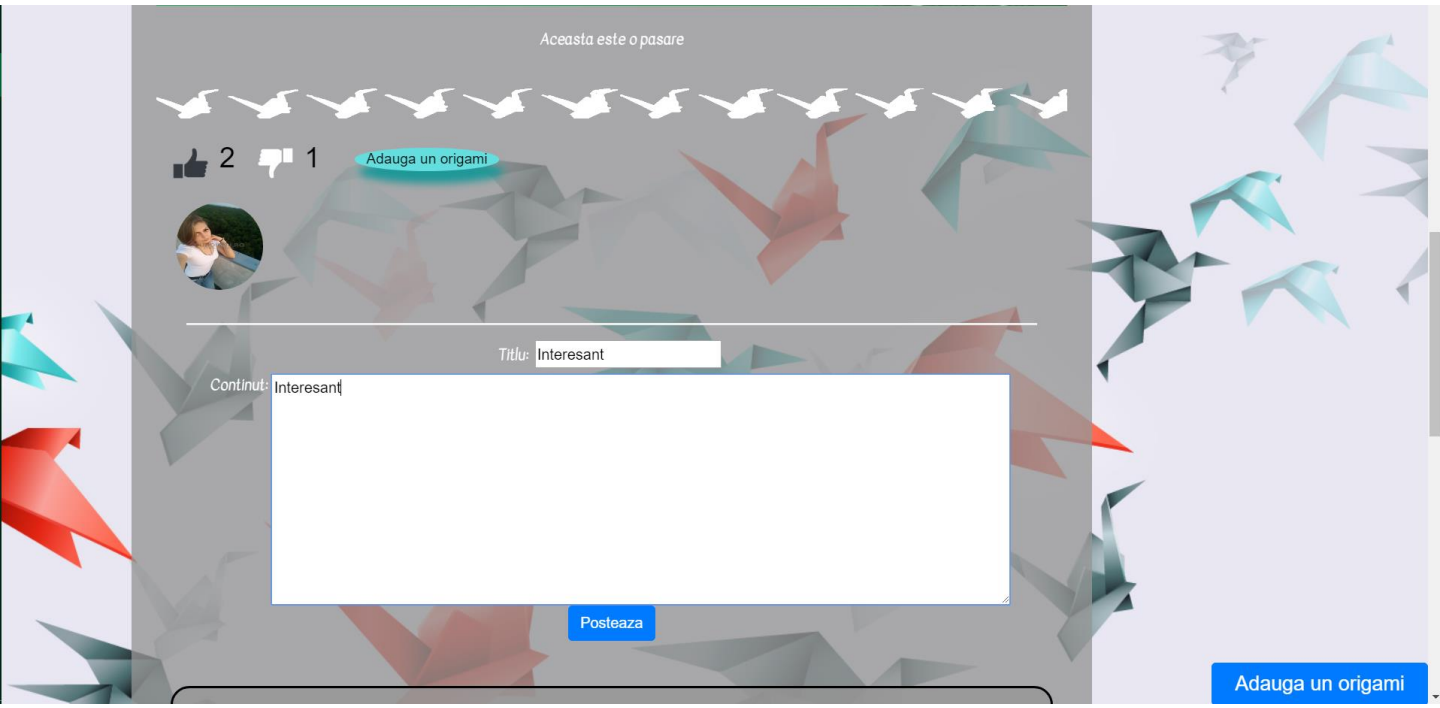



```
      <input type="text" class="form-control" name="username"  ng-model="user.username" id="exampleInputUsername"
aria-describedby="userHelp" placeholder="Enter user">
    </div>
    <div class="form-group">
      <label for="exampleInputNume">Nume</label>
      <input type="text" class="form-control" name="name"  id="exampleInputNume"
aria-describedby="numeHelp" placeholder="Enter name"  ng-model="user.name">
    </div>
    <div class="form-group">
      <label for="exampleInputPrenume">Prenume</label>
      <input type="text" class="form-control" name="surname" id="exampleInputPrenume"
aria-describedby="emailHelp" placeholder="Enter email"  ng-model="user.surname">
    </div>
    <div class="form-group">
      <label for="exampleInputPassword1">Pariola     <span class="tooltip1"><img src="alerta.png" class="alerta"><span class="tooltiptext">Introduceti o adresa de email valida</span></span>
      </label>
      <input type="password" name="password"  class="form-control"
id="exampleInputPassword1" placeholder="Password"  ng-model="user.password">
    <small class="form-text text-muted">Nu vom dezvalui parola nimanui</small>
    </div>
    <div>
   <p>Sexul: <span class="radioValue">{{ user.gender }}</span> </p>
    <md-radio-group  name="sex"  ng-model="user.gender">
      <md-radio-button value="0" class ="md-primary">Masculin</md-radio-button>
      <md-radio-button value="1"> Feminin </md-radio-button>
    </md-radio-group> </div>
    <p>Data nasterii:
      <md-datepicker ng-model="user.birthdate" name="date"  md-current-view="year"
md-placeholder="Enter date" onkeydown="return false"></md-datepicker>
    </p>
    <button type="submit" class="btn btn-primary">Trimite</button>
  </div>
</form>
</div>
 <div class="col-4"></div>
</div>

<footer style="position: absolute;width:100%;background-color:rgba(150,150,150,0.78);margin:5% 0 0 0;font-size: 1vw;">
      <div id="stanga">
          <p style="margin: 0;">Creatori: Iulia Corcodel, Colegiul National de Informatica "Tudor Vianu", clasa a 11-a si <br> Andrei Gavrea, Colegiul National "Mihai Eminescu", clasa a 10-a</p>&copy
          ;Origami<br>
            <a href="https://github.com/iuliaco/Origami" target="_blank">github</a>
        </div> <div id="dreapta">
          <table align="center">
              <tr><td>Email:</td><td><a href="mailto:andy7035@yahoo.com" target="_top">andy7035@yahoo.com</a></td></tr>
                <tr><td></td><td><a href="mailto:iuliaco2001@gmail.com" target="_top">iuliaco2001@gmail.com</a></td></tr>
            </table>
        </div>
    </footer>
<script>

var app = angular.module("ori", ['ngMaterial']);
```

Line 139, Column 66 Spaces: 2 HTML

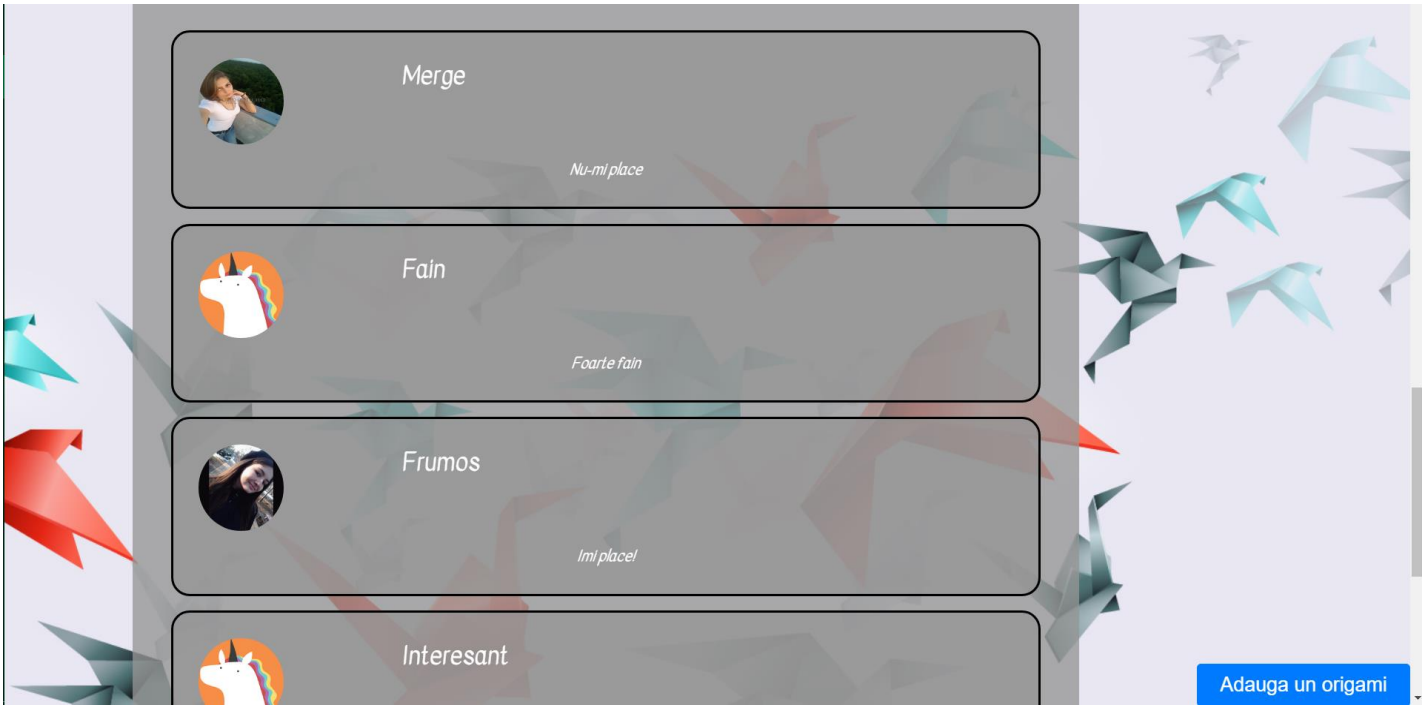
Pe pagina profilului tau vei putea sa vezi datele introduse la completarea formularului, data crearii contului, dar si imaginea ta de profil, pe care o poti schimba cand doresti.
Doar daca esti autentificat poti aprecia alte modele de origami si poti adauga comentarii la acestea.





Utilizatorii pot adauga comentarii si aprecia postarile de pe site. Sectiunea de comentarii este separata de restul continutului prin utilizarea clasei “comment” care ii ofera divului o bordura in partea de sus.

Comentariile facute anterior sunt despartite prin bordure.



Origami Acasa Inregistrare/Autentificare

Lup origami

Origami Acasa Inregistrare/Autentificare

LOG IN REGISTER

Adresa de email/Numele de utilizator

Email/Nume de utilizator

Parola

Parola

Submit

Origami Acasa Inregistrare/Autentificare

LOG IN REGISTER

Adresa de email

andy7035@yahoo.com

Nume de utilizator

Mr.

Numele de familie

Nume

Prenume

Prenume

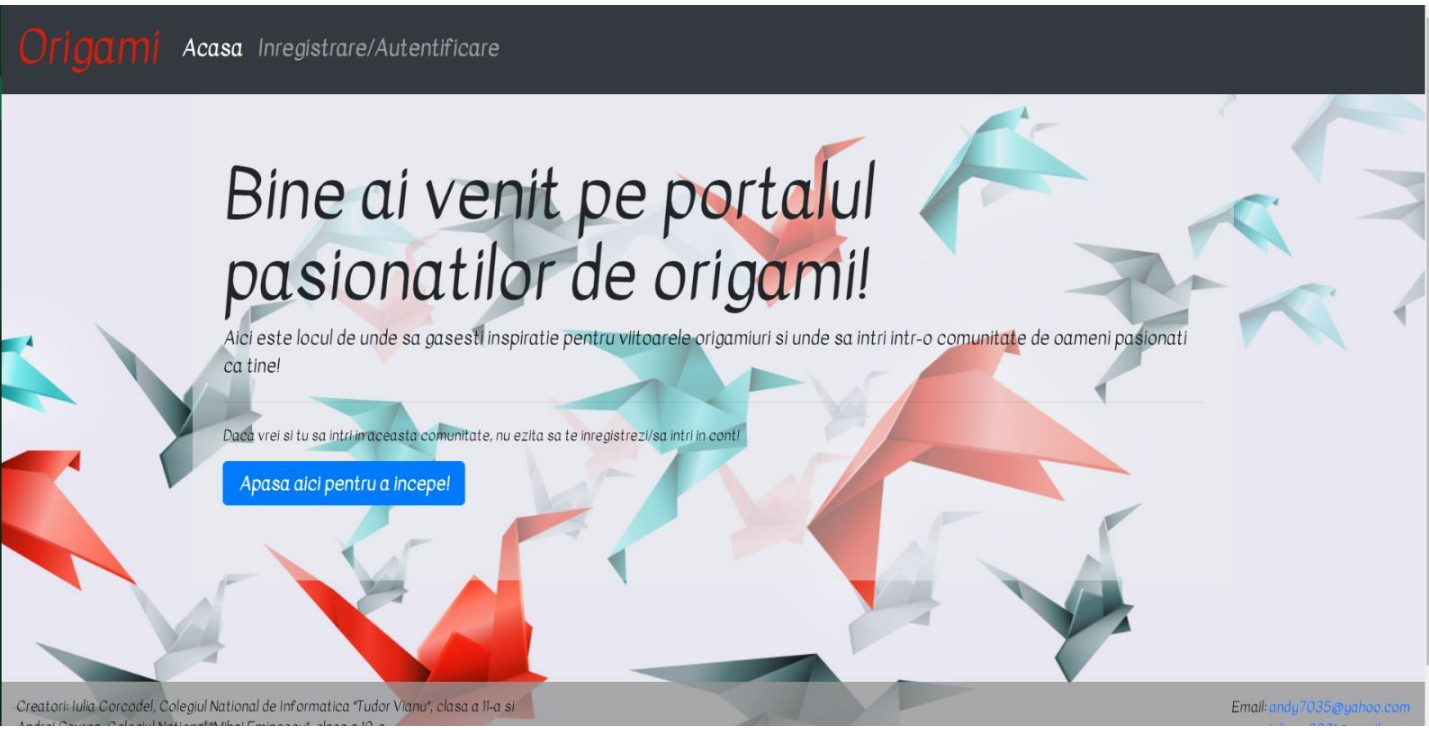
Parola

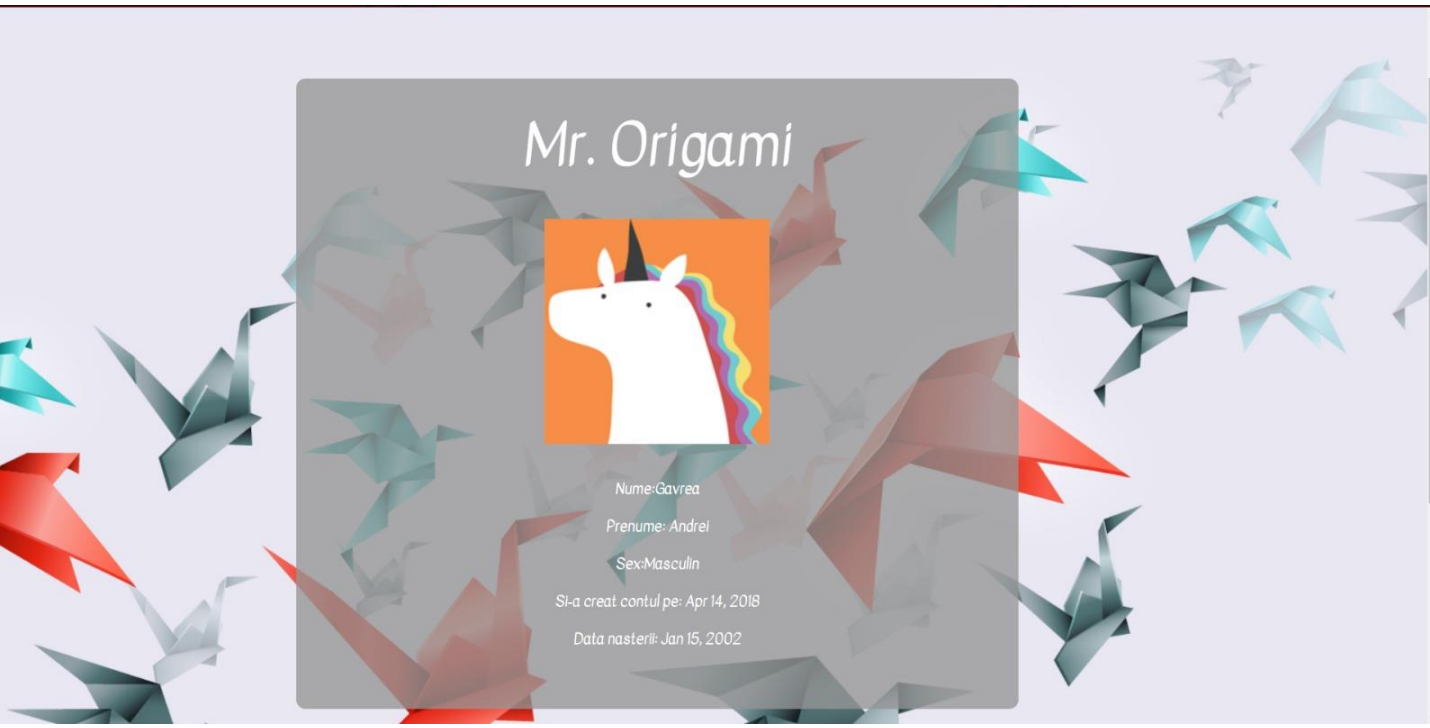
••

Sexul:

Masculin

Feminin





Aceasta este poza de default la profil pentru cei care nu au una.