

**UNIVERSITATEA TEHNICĂ A MOLDOVEI**  
**FACULTATEA CALCULATOARE, INFORMATICĂ ȘI**  
**MICROELECTRONICĂ**

**UTILIZAREA TEHNOLOGIILOR HTML,**  
**XHTML, CSS, LIMBAJE CLIENT SIDE,**  
**LIMBAJE SERVER SIDE, APLICAȚII RIA,**  
**TEHNOLOGIA AJAX LA REALIZAREA UNEI**  
**APLICAȚII WEB MULTIMEDIA**

**LUCRARE DE LABORATOR NR. 5**

**la disciplina ”Sisteme multimedia”**

Autor:

studentul gr. TI 141,

învățăământ cu frecvență redusă

MIROVSCHI Eugen

Profesor:

Lector superior,

SAVA Nina

---

(semnătura)

**Chișinău, 2018**

## I. Scopul lucrării

Familiarizarea cu limbajele HTML, XHTML, CSS, Javascript, Java.

## II. Sarcina lucrării

Realizarea unei aplicații web multimedia utilizând noțiunile din HTML, XHTML, CSS. Să se utilizeze limbaje client side și server side.

## III. Efectuarea lucrării

Pentru a crea o aplicație web multimedia am decis să folosesc platforma Java ca server side. Respectiv proiectul a fost creat utilizând IntelliJ ca IDE și Maven ca build manager. Maven oferă posibilitatea de a adăuga automat toate dependențele inclusiv cele tranzitive. Pentru a crea o aplicație web este necesar de a crea un web server. Acest lucru a fost obținut prin adăugarea framework-ului spring și modulul boot. Acesta oferă un sistem de dependency injection care creează posibilitatea de a adăuga componente și de a suplini dependențele acestora. O altă parte necesară din acest framework este integrarea pattern-ului MVC care permite crearea unui serviciu REST API.

```
@Component("postConverter")
public class PostConverter implements Converter<PostModel, PostDTO>
{
    @Resource
    private Populator<PostModel, PostDTO> postMinimalPopulator;

    @Resource
    private Populator<PostModel, PostDTO> postVotesPopulator;

    @Override
    public PostDTO convert(final PostModel post)
    {
        final PostDTO postDTO = new PostDTO();
        postMinimalPopulator.populate(post, postDTO);
        postVotesPopulator.populate(post, postDTO);
        return postDTO;
    }
}
```

Figura 1 – Exemplu de crearea a unei componente spring

```

@RestController
@RequestMapping("/tags")
public class TagController
{
    @Autowired
    private TagFacade tagFacade;

    @RequestMapping(value = "/search", method = RequestMethod.POST)
    public ListDTO<TagDTO> searchTags(@RequestBody final SearchDTO search)
    {
        return tagFacade.searchTags(search);
    }
}

```

**Figura 2 – Definirea de REST endpoint**

Spring oferă posibilitatea de a defini entități folosind JPA framework și repozitorii care vor gestiona aceste entități. Astfel acest framework permite implementarea proiectului fără a depinde de o anumită bază de date. În acest caz am utilizat local baza de date de tip H2. În momentul când voi dori să trec pe alt server cu alt tip de bază de date, nu voi avea nicio dificultate.

```

@Entity
@Table(name = "tags")
public class TagModel
{
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator = "tags_id_seq")
    @GenericGenerator(name = "tags_id_seq", strategy = "org.hibernate.id.enhanced.SequenceStyleGenerator")
    private long id;

    @Column(unique = true)
    private String name;

    public String getName() { return name; }

    public void setName(final String name) { this.name = name; }
}

```

**Figura 3 – Exemplu de entitate**

```

@Repository
public interface TagRepository extends JpaRepository<TagModel, Long>
{
    TagModel findByName(String name);

    @Query(SEARCH_TAGS)
    List<TagModel> findTags(@Param("query") String query, @Param("addsCount") long addsCount,
        @Param("adds") Set<TagModel> adds, @Param("excludes") Set<TagModel> excludes);
}

```

**Figura 4 – Exemplu de repozitoriu**

O altă funcționalitate importantă oferită de acest framework este Spring Boot care include în sine HTTP server și elimină dependența de careva aplicație gen Tomcat.

Partea front-end a fost implementată utilizând framework-ul AngularJS. Aceasta oferă posibilitatea de a crea un single page application utilizând MVW pattern.

```
app.controller("login", function($scope, $mdDialog, auth)
{
    $scope.request = auth.request;

    $scope.isLoading = function()
    {
        return auth.isLoading();
    }

    $scope.close = function()
    {
        $mdDialog.hide();
    }

    $scope.login = function()
    {
        {
            auth.login(function()
            {
                $scope.close();
            },
            function(data)
            {
                $scope.loginForm["username"].$setValidity("bad", false);
                $scope.loginForm["password"].$setValidity("bad", false);
            });
        }
    }
});
```

**Figura 5 – Exemplu de controller în AngularJS**

```

<md-dialog flex="90" flex-gt-xs="75" flex-gt-sm="50" flex-gt-md="30">
  <form name="loginForm" role="form" ng-submit="login()" ng-cloak>
    <md-toolbar class="md-primary md-hue-2" md-theme="login">
      <div class="md-toolbar-tools">
        <h2>Log In</h2>
        <span flex></span>
        <md-button class="md-icon-button" ng-click="close()">
          <md-icon<i class="material-icons">close</i></md-icon>
        </md-button>
      </div>
    </md-toolbar>

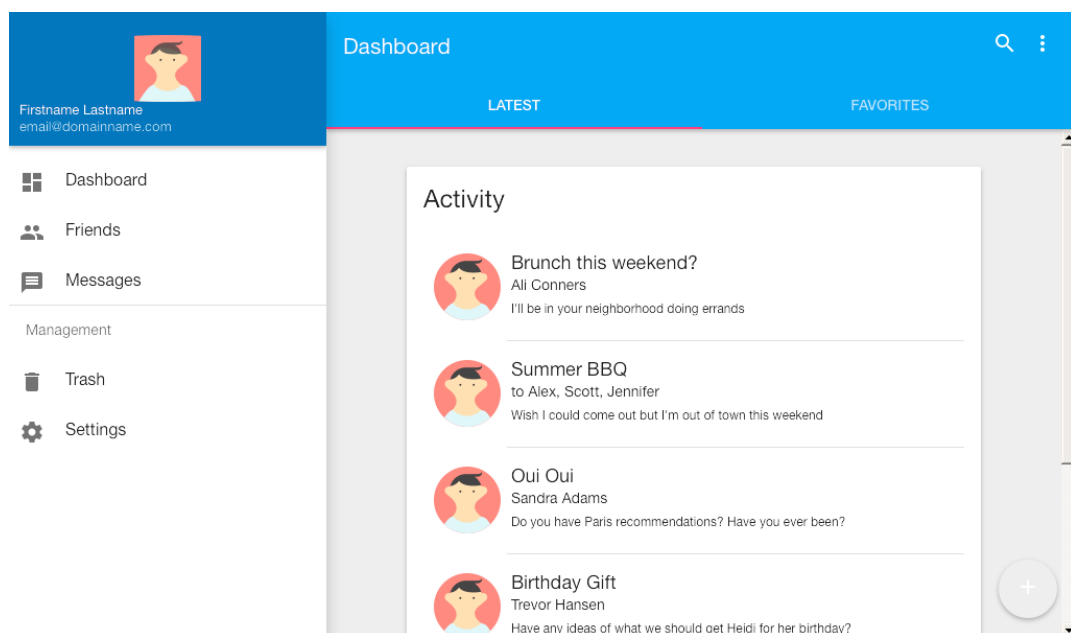
    <md-dialog-content>
      <div class="md-dialog-content" layout="column" layout-align="start stretch">
        <md-input-container class="md-block">
          <label>Email</label>
          <input ng-model="request.username" type="text" name="username" ng-disabled="isLoading()" />
          <div ng-messages="loginForm.password.$error">
            <div ng-message="bad"></div>
          </div>
        </md-input-container>
        <md-input-container class="md-block">
          <label>Password</label>
          <input ng-model="request.password" type="password" name="password" autocomplete="new-password" ng-disabled="isLoading()" />
          <div ng-messages="loginForm.password.$error">
            <div ng-message="bad">Incorrect username or password</div>
          </div>
        </md-input-container>
      </div>
    </md-dialog-content>

    <md-dialog-actions>
      <md-button flex class="md-flex md-primary md-raised md-hue-2" ng-disabled="isLoading()" type="submit">Log In</md-button>
    </md-dialog-actions>
  </form>
  <md-progress-linear md-mode="intermediate" ng-show="isLoading()"></md-progress-linear>
</md-dialog>

```

**Figura 6 – Exemplu de șablon HTML**

Pentru ușura definirea elementelor a fost inclusă extensia Angular Material care oferă un set de componente web, instrucțiuni de definire a structurii pagini și a stilurilor acesteia inclusiv pentru diferite dispozitive.



**Figura 7 – Exemplu de aplicație făcută folosind Material**

După implementarea părții server și cea client am obținut o aplicație web care permite încărcarea și publicarea fișierelor media cum ar fi imagini, audio și video. Exemple de pagini pot fi văzute în figurile de mai jos.

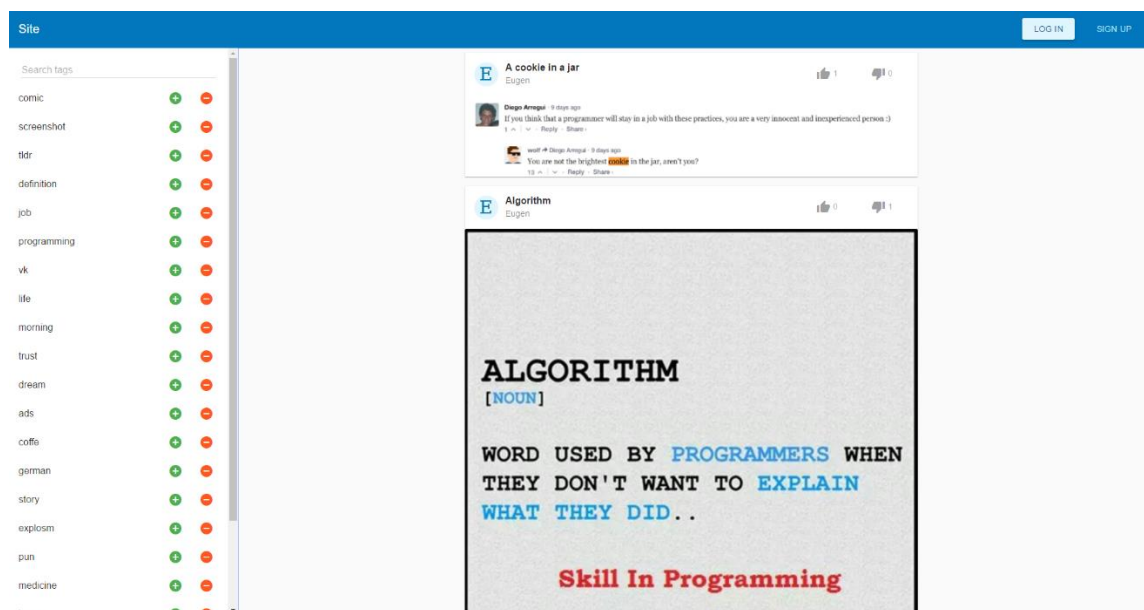


Figura 8 – Exemplu de pagină principală

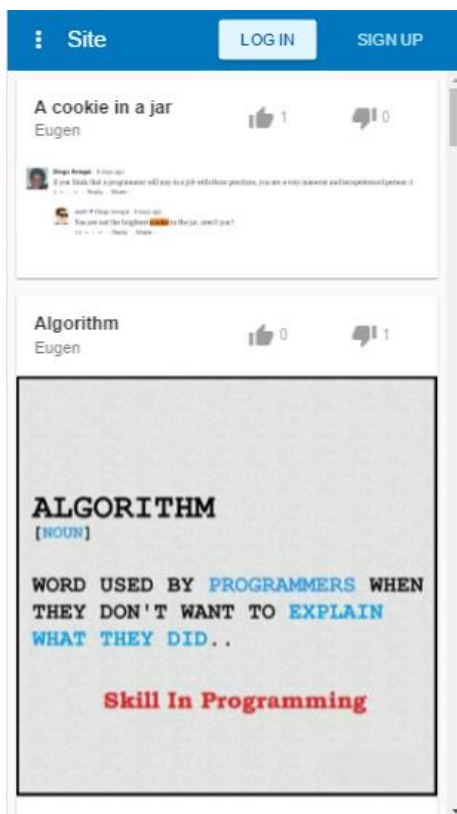


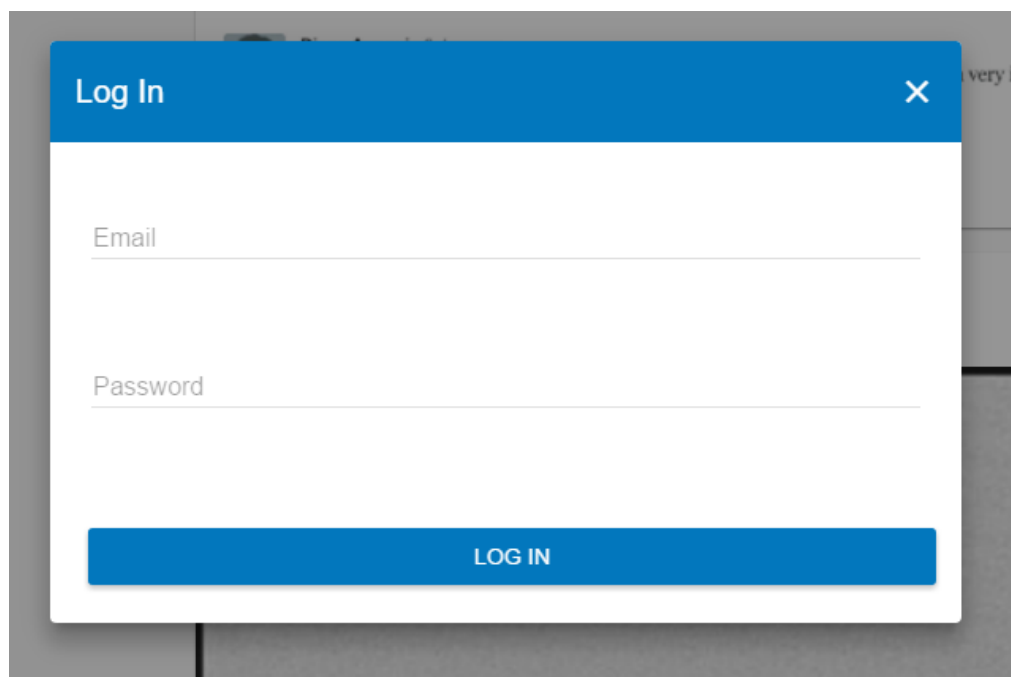
Figura 9 - Exemplu de pagină principală vizibilă de pe un dispozitiv mobil



**Figura 10 – Exemplu de sidebar de pe dispozitiv mobil**

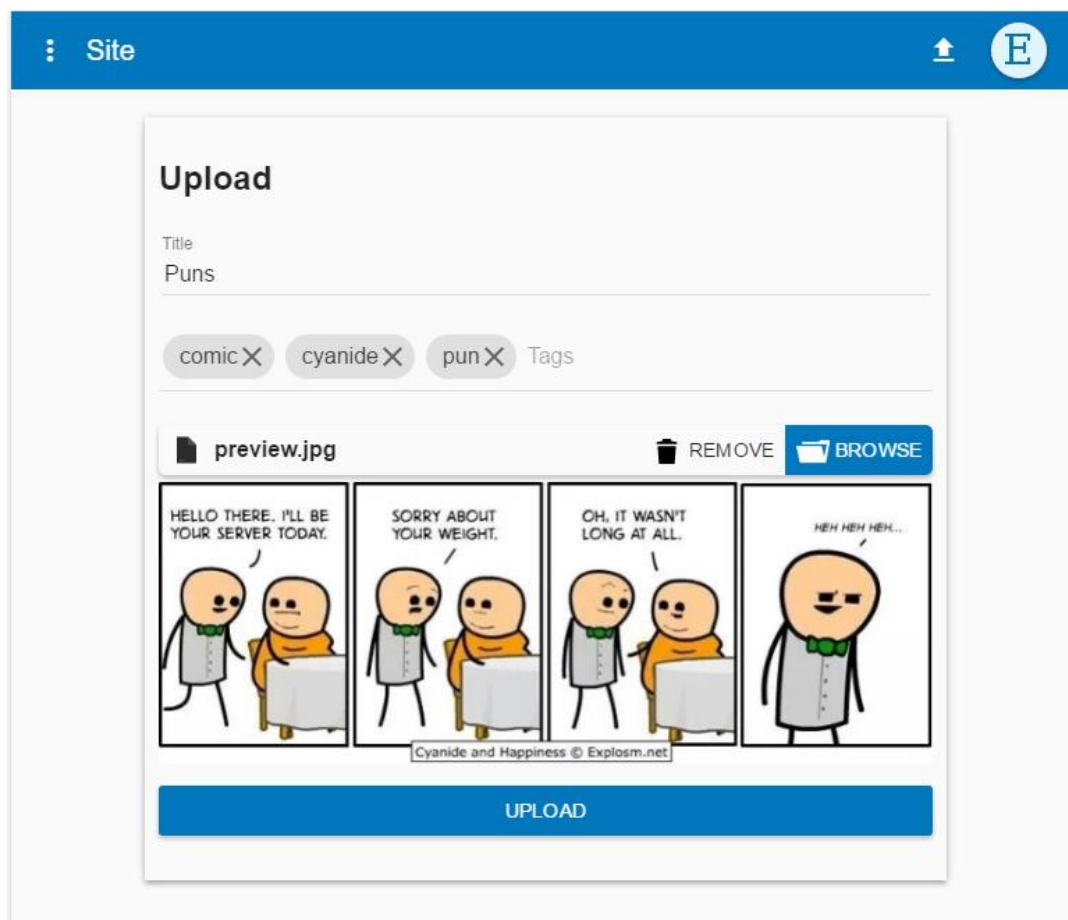
A mobile sign-up form titled "Sign Up" with a close button (X) in the top right corner. The form contains four input fields: "Name", "Email", "Password", and "Repeat password". At the bottom of the form is a blue button labeled "SIGN UP".

**Figura 11 – Pagina de înregistrare**



A modal window titled "Log In" with a close button (X) in the top right corner. It contains two input fields: "Email" and "Password". At the bottom, there is a blue button labeled "LOG IN".

Figura 12 – Pagina de autentificare



A web page titled "Site" with a blue header bar. The main content area is titled "Upload". It includes a "Title" field and a "Puns" field. Below these are three tags: "comic", "cyanide", and "pun", each with a close button (X). To the right of the tags is a "Tags" label. Below the tags is a preview image of a comic strip titled "preview.jpg". The comic strip shows two characters, one in a white lab coat and one in an orange robe, sitting at a table. The lab coat character says: "HELLO THERE, I'LL BE YOUR SERVER TODAY.", "SORRY ABOUT YOUR WEIGHT.", "OH, IT WASN'T LONG AT ALL.", and "HEH HEH HEH...". The orange robe character says: "HEH HEH HEH...". Below the comic strip is a blue button labeled "UPLOAD".

Figura 13 – Pagina de încărcare a unui fișier media



Site

E

Account settings

Name

Eugen

Password

Repeat password

UPDATE

**Figura 14 – Pagina de configurare a contului**

## **IV. Concluzii**

Pentru efectuarea acestei lucrări am utilizat un set de instrumente care m-au ajutat la implementarea unei aplicații web. Ca mediu de dezvoltare am folosit IntelliJ care permite redactarea optimă a codului Java și are integrarea cu Maven. Pentru a controla structura și modul de construire a proiectului, precum și includerea eficientă și rapidă a dependențelor externe am folosit managerul Maven. Acest site este compus din 2 părți principale: Backend și Frontend.

Prima este scrisă în Java și conține toată logica care se execută pe partea de server side. Una din dependențele principale utilizate aici este Spring framework care ușurează crearea componentelor injectând automat toate dependențele, permite crearea unui REST API folosind controllere și metode anotate, oferă posibilitatea de a integra o bază de date într-un mod generic care ulterior permite să aplici orice tip de bază de date relațională.

Pentru partea de frontend am folosit 2 framework-uri: AngularJS și Angular Material care conlucrează între ele. Primul oferă posibilitatea de a crea un single page application folosind MVW pattern. Angular Material oferă un set de utilități care ușurează procesul de a crea toate elementele din pagină și permite reutilizarea acestora cu scopul de a defini structura paginii pentru mai multe dispozitive. Toate aceste instrumente mi-au permis să dezvolt eficient un produs software calitativ.