**VISOKA TEHNIČKA ŠKOLA**

**STRUKOVNIH STUDIJA**

**S** **U B O T I C A**

**PROJEKAT**

**iz predmeta Web programiranje**

|  |  |
| --- | --- |
| **KANDIDAT(I)**  **Tabajica Ognjen** | **MENTOR**  **prof. dr Zlatko Čović** |
| **SUBOTICA, 2025.** | |

Sadržaj

[1. Opis zadatka 3](#_Toc156903284)

[2. Realizacija zadatka 4](#_Toc156903285)

[3. Struktura baze podataka (MySQL Workbench) 5](#_Toc156903286)

[4. Opis funkcionalnosti 6](#_Toc156903287)

[5. Korišćena literatura 7](#_Toc156903288)

# Opis zadatka

**Cilj zadatka je da se kreira web aplikaciju za bezu poslova. Aplikacija razlikuje četiri nivoa pristupa: Gost, prijavljeni korisnik, prijavljena kompanija, administrator**

**Gost ima privilegiju da vidi oglase, i da ih preko kategorije filtrira, ali ne i da se na njih prijavi**

**Prijavljeni korisnik ima dodatne privilegije da se na oglase za posao prijavi, takođe može da menja osnovne lične informacije na stranici Profile**

**Prijavljena kompanija može da postavlja oglase za posao kao i da vidi prijavljene korisnike i da ih putem maila kontaktira**

**root**

**│ .env**

**│ .gitattributes**

**│ .gitignore**

**│ .htaccess**

**│ composer.json**

**│ composer.lock**

**│ config.php**

**│ database.sql**

**│ db.php**

**│ index.php**

**│ init.php**

**│ login.php**

**│ logout.php**

**│ profile.php**

**│ register.php**

**│ tree.txt**

**│**

**├───admin**

**│ entities.php**

**│ listings.php**

**│ login.php**

**│ logout.php**

**│**

**├───api**

**│ │ apply.php**

**│ │ categories.php**

**│ │ listings.php**

**│ │ login.php**

**│ │ logout.php**

**│ │ register.php**

**│ │**

**│ ├───admin**

**│ │ categories.php**

**│ │ category\_create.php**

**│ │ category\_delete.php**

**│ │ companies.php**

**│ │ company\_delete.php**

**│ │ company\_get.php**

**│ │ company\_save.php**

**│ │ jobs.php**

**│ │ job\_delete.php**

**│ │ job\_get.php**

**│ │ job\_save.php**

**│ │ login.php**

**│ │ users.php**

**│ │ user\_delete.php**

**│ │ user\_get.php**

**│ │ user\_save.php**

**│ │ \_require\_admin.php**

**│ │**

**│ ├───company**

**│ │ applicants.php**

**│ │ categories.php**

**│ │ contact.php**

**│ │ create\_job.php**

**│ │ delete\_job.php**

**│ │ login.php**

**│ │ my\_jobs.php**

**│ │ register.php**

**│ │**

**│ └───user**

**│ login.php**

**│**

**├───assets**

**│ admin\_entities.js**

**│ admin\_listings.js**

**│ company.js**

**│ logout.js**

**│ main.js**

**│ styles.css**

**│**

**├───company**

**│ index.php**

**│ login.php**

**│ register.php**

**│**

**│**

**├───lib**

**email.php**

**ensure\_admins.php**

**ensure\_companies.php**

# Realizacija zadatka

**Main.js**

Javascript fajl zaduzen za funkcionisanje index.php I ostalih fajlova u root-u projekta (gosti i korisnici)

async function fetchCategories() {

  const r = await fetch("api/categories.php", {

    headers: { Accept: "application/json" },

  });

  const j = await r.json();

  if (!j.ok) throw new Error(j.error || "Failed to load categories");

  return j.categories;

}

async function fetchJobs(params = {}) {

  const qs = new URLSearchParams();

  if (params.category\_id && Number(params.category\_id) > 0)

    qs.set("category\_id", params.category\_id);

  const url = "api/listings.php" + (qs.toString() ? "?" + qs.toString() : "");

  const res = await fetch(url, { headers: { Accept: "application/json" } });

  const data = await res.json();

  if (!data.ok) throw new Error(data.error || "Failed to load jobs");

  return data.jobs;

}

function jobCard(job) {

  const div = document.createElement("div");

  div.className = "card job";

  div.innerHTML = `

    <div class="title">${escapeHtml(job.title)}</div>

    <div class="meta">

      <span class="tag">${escapeHtml(job.category || "General")}</span>

      <span class="tag">${escapeHtml(job.city)}</span>

      <span class="tag">${escapeHtml(job.company\_name)}</span>

    </div>

    <div class="desc">${escapeHtml(job.description)}</div>

    <button class="btn primary block apply-btn" data-id="${

      job.id

    }">Apply</button>

  `;

  const btn = div.querySelector(".apply-btn");

  // Not logged in -> disabled

  if (!window.IS\_LOGGED\_IN) {

    btn.setAttribute("disabled", "disabled");

    btn.title = "Please login to apply";

    return div;

  }

  // Logged in: if already applied, mark & disable

  if (Number(job.has\_applied) === 1) {

    btn.textContent = "Applied ✓";

    btn.classList.add("applied");

    btn.setAttribute("disabled", "disabled");

  } else {

    btn.addEventListener("click", () => apply(job.id, btn));

  }

  return div;

}

async function apply(jobId, button) {

  button.disabled = true;

  try {

    const res = await fetch("api/apply.php", {

      method: "POST",

      headers: {

        "Content-Type": "application/json",

        Accept: "application/json",

      },

      body: JSON.stringify({ job\_id: jobId }),

    });

    const data = await res.json();

    if (!res.ok || !data.ok) throw new Error(data.error || "Apply failed");

    // reflect applied state immediately

    button.textContent = "Applied ✓";

    button.classList.add("applied");

    button.disabled = true;

  } catch (e) {

    alert(e.message);

    button.disabled = false;

  }

}

function escapeHtml(s) {

  return String(s ?? "").replace(

    /[&<>"']/g,

    (c) =>

      ({ "&": "&amp;", "<": "&lt;", ">": "&gt;", '"': "&quot;", "'": "&#39;" }[

        c

      ])

  );

}

async function init() {

  const jobsContainer = document.getElementById("jobs");

  const select = document.getElementById("category-filter");

  // Populate categories

  try {

    const cats = await fetchCategories();

    select.innerHTML =

      `<option value="0">All categories</option>` +

      cats

        .map((c) => `<option value="${c.id}">${escapeHtml(c.name)}</option>`)

        .join("");

  } catch (e) {

    select.innerHTML = `<option value="0">(failed to load)</option>`;

  }

  // Load and render jobs (with optional category filter)

  async function renderJobs() {

    jobsContainer.innerHTML = '<p class="muted">Loading...</p>';

    try {

      const jobs = await fetchJobs({ category\_id: select.value });

      if (!jobs.length) {

        jobsContainer.innerHTML = '<p class="muted">No jobs found.</p>';

        return;

      }

      const frag = document.createDocumentFragment();

      jobs.forEach((j) => frag.appendChild(jobCard(j)));

      jobsContainer.innerHTML = "";

      jobsContainer.appendChild(frag);

    } catch (e) {

      jobsContainer.innerHTML = `<div class="alert error">${escapeHtml(

        e.message

      )}</div>`;

    }

  }

  select.addEventListener("change", renderJobs);

  await renderJobs();

}

init();

logout.js funkcijonalnost logout dugmeta radi redirect-ovanja u project root

document.addEventListener('DOMContentLoaded', () => {

  const btn = document.getElementById('logoutBtn');

  if (!btn) return;

  btn.addEventListener('click', async (e) => {

    e.preventDefault();

    const url = window.LOGOUT\_URL || 'api/logout.php';

    try {

      await fetch(url, { method: 'POST' });

    } finally {

      location.href = window.LOGOUT\_REDIRECT || 'https://studcp.vts.su.ac.rs:10000/virtual-server/link.cgi/147.91.199.133/http://www.evasion.stud.vts.su.ac.rs/';

    }

  });

});

### login.php

<?php

// api/login.php

require\_once \_\_DIR\_\_ . '/../init.php';

header('Content-Type: application/json; charset=utf-8');

$email = trim($\_POST['email'] ?? '');

$password = $\_POST['password'] ?? '';

if ($email === '' || $password === '') {

  http\_response\_code(400);

  echo json\_encode(['ok' => false, 'error' => 'Email and password required']);

  exit;

}

$st = get\_pdo()->prepare('SELECT id, name, email, password\_hash FROM users WHERE email = ? LIMIT 1');

$st->execute([$email]);

$u = $st->fetch();

if ($u && password\_verify($password, $u['password\_hash'])) {

  $\_SESSION['user'] = ['id' => $u['id'], 'name' => $u['name'], 'email' => $u['email']];

  echo json\_encode(['ok' => true]);

  exit;

}

http\_response\_code(401);

echo json\_encode(['ok' => false, 'error' => 'Invalid credentials']);

## api/company/register.php

<?php

// api/company/register.php

require\_once \_\_DIR\_\_ . '/../../init.php';

require\_once \_\_DIR\_\_ . '/../../lib/ensure\_companies.php';

header('Content-Type: application/json; charset=utf-8');

ensure\_companies\_table(get\_pdo());

$name = trim($\_POST['name'] ?? '');

$email = trim($\_POST['email'] ?? '');

$password = $\_POST['password'] ?? '';

$password2 = $\_POST['password2'] ?? '';

$website = trim($\_POST['website'] ?? '');

$address = trim($\_POST['address'] ?? '');

$about = trim($\_POST['about'] ?? '');

if ($name === '' || $email === '' || $password === '' || $password2 === '') {

  http\_response\_code(400);

  echo json\_encode(['ok' => false, 'error' => 'Required fields: name, email, password']);

  exit;

}

if (!filter\_var($email, FILTER\_VALIDATE\_EMAIL)) {

  http\_response\_code(400);

  echo json\_encode(['ok' => false, 'error' => 'Invalid email']);

  exit;

}

if ($password !== $password2) {

  http\_response\_code(400);

  echo json\_encode(['ok' => false, 'error' => 'Passwords do not match']);

  exit;

}

$pdo = get\_pdo();

$st = $pdo->prepare('SELECT id FROM companies WHERE email = ? OR name = ? LIMIT 1');

$st->execute([$email, $name]);

if ($st->fetch()) {

  http\_response\_code(409);

  echo json\_encode(['ok' => false, 'error' => 'Company email or name already registered']);

  exit;

}

$hash = password\_hash($password, PASSWORD\_DEFAULT);

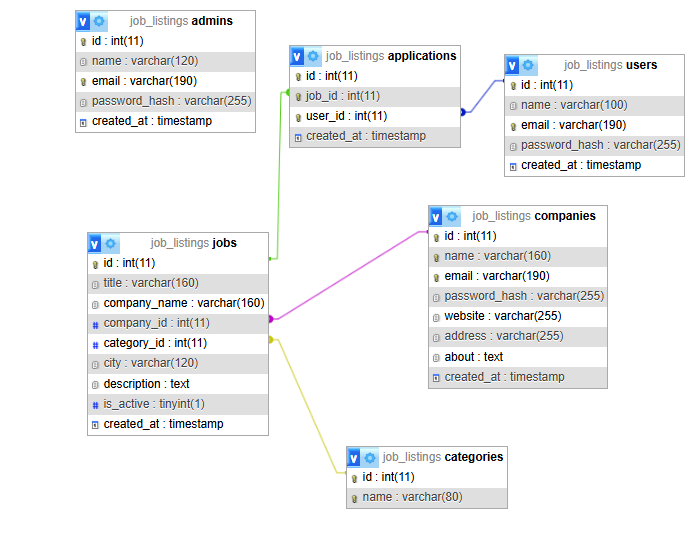
$ins = $pdo->prepare('INSERT INTO companies (name,email,password\_hash,website,address,about) VALUES (?,?,?,?,?,?)');

$ins->execute([$name,$email,$hash,$website,$address,$about]);

$\_SESSION['company'] = ['id' => $pdo->lastInsertId(), 'name' => $name, 'email' => $email];

echo json\_encode(['ok' => true]);

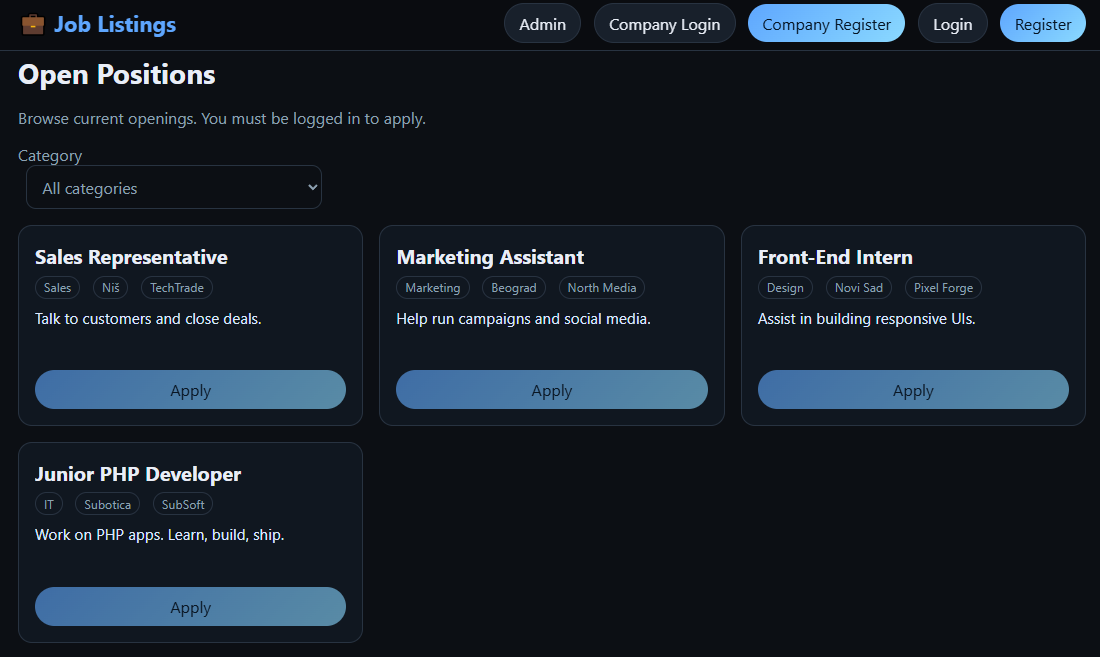
# Struktura baze podataka (MySQL Workbench)

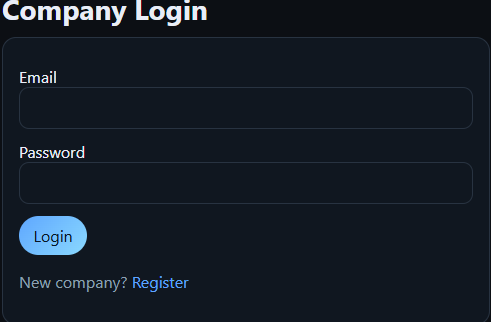
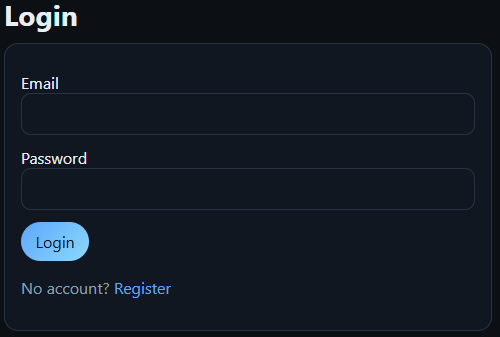
****

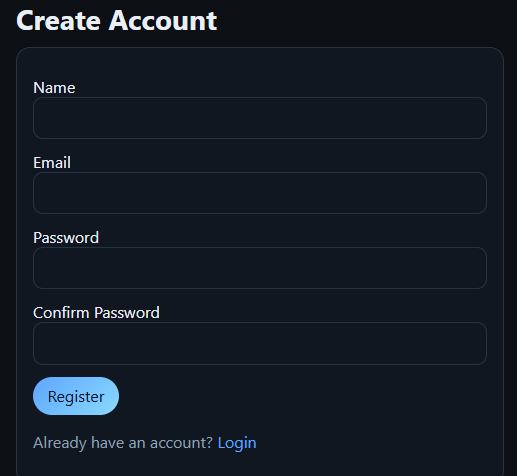
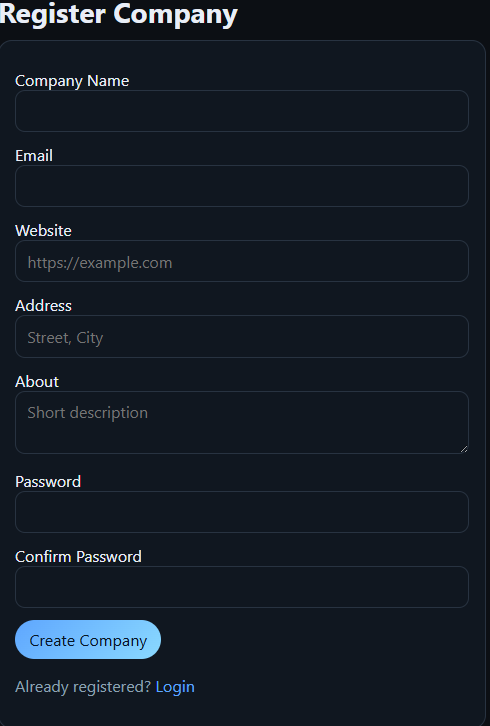
# Opis funkcionalnosti

U ovom delu obavezno opisati način funkcionalnosti projekta. Koristiti slike (*screenshots*).

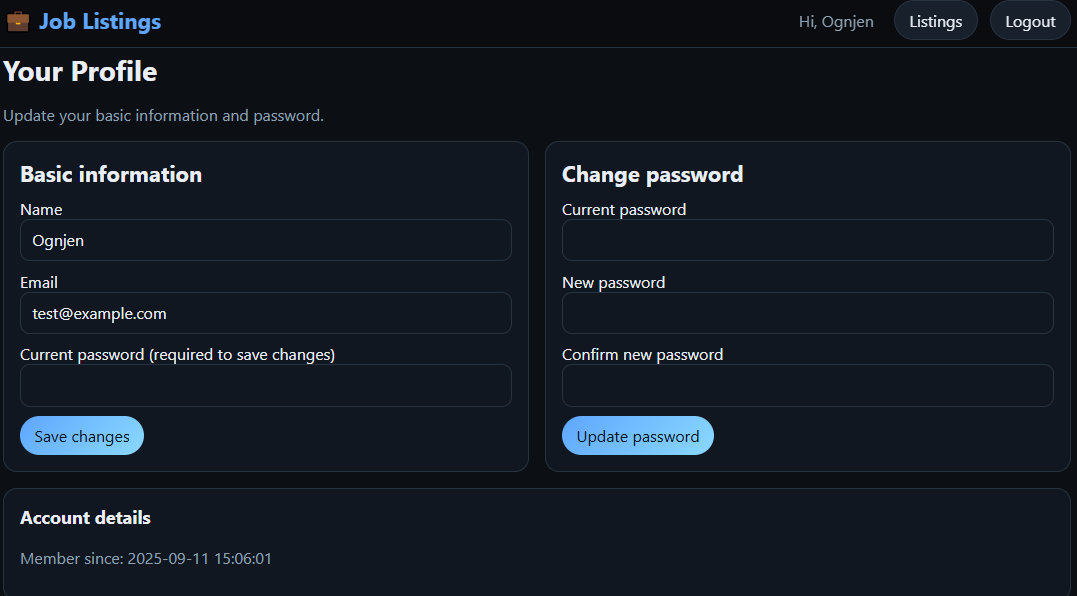
**Početna stranica, gosti mogu da vide sve trenotno aktivne poslove i da ih filtriraju po kategorijama (odnosno po njihovoj struci) mogu da se registruju i loginuju na sajt.**

****

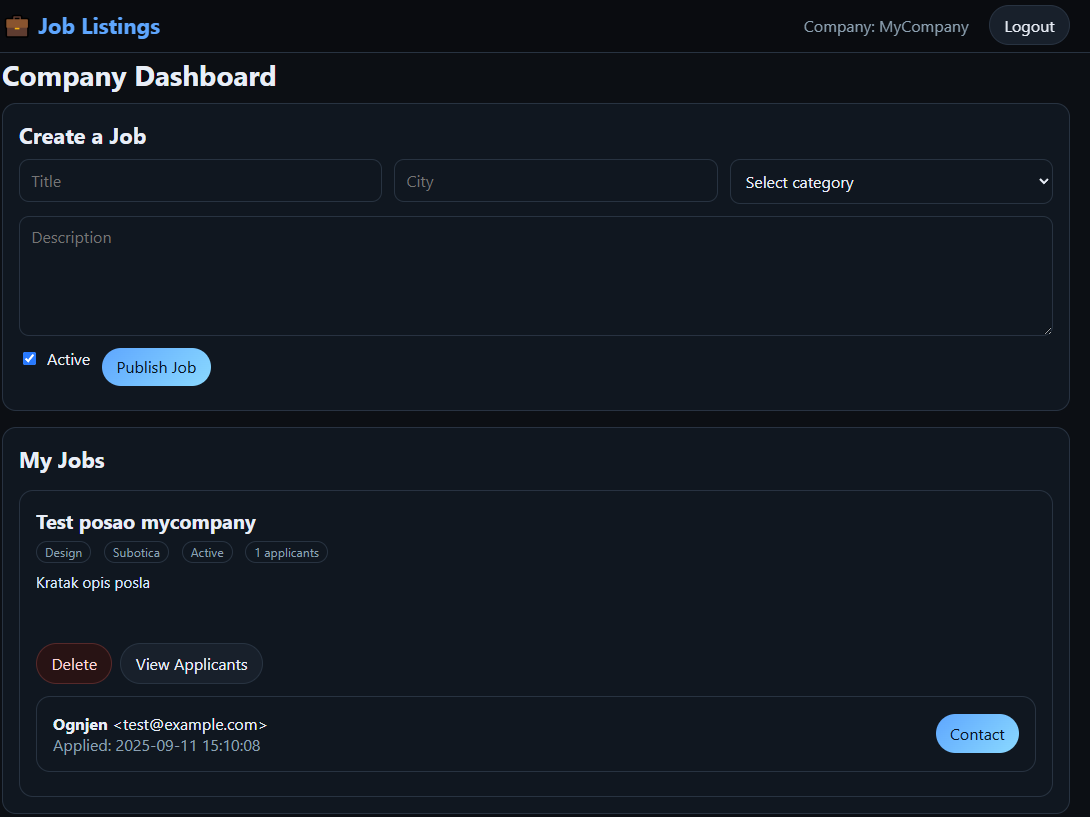
****

****

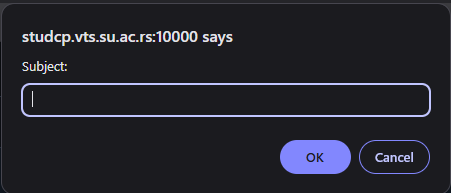
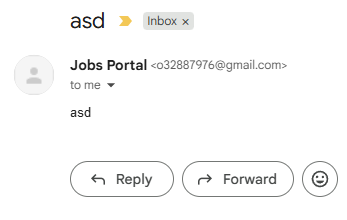
**Korisnici kada se registruju/prijave omogućeno im je da se prijave na oglas za posao, takođe imaju pristup svom profilu gde mogu da menjaju svoje podatke**

****

Kada se korisnik prijavi na oglas njegove informacije su prikazane kompaniji koja je oglas kreirala.

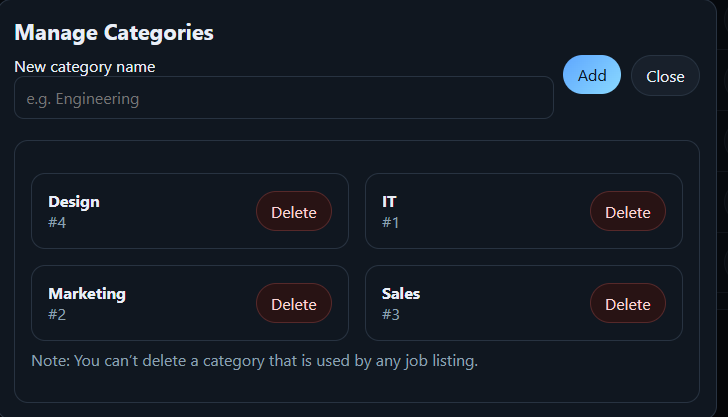
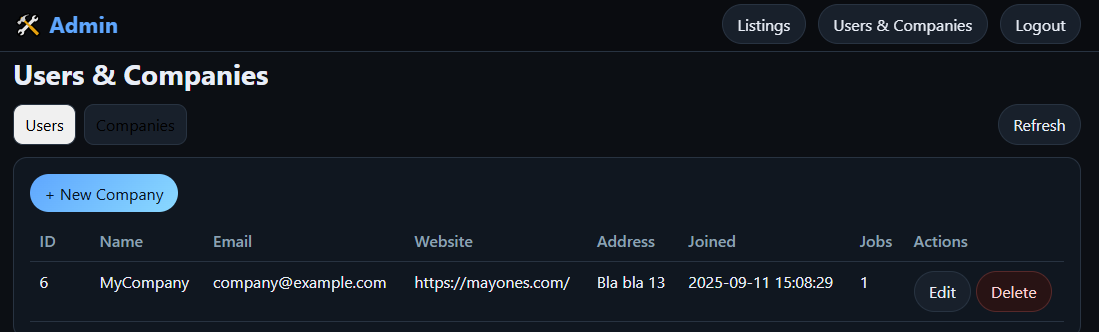
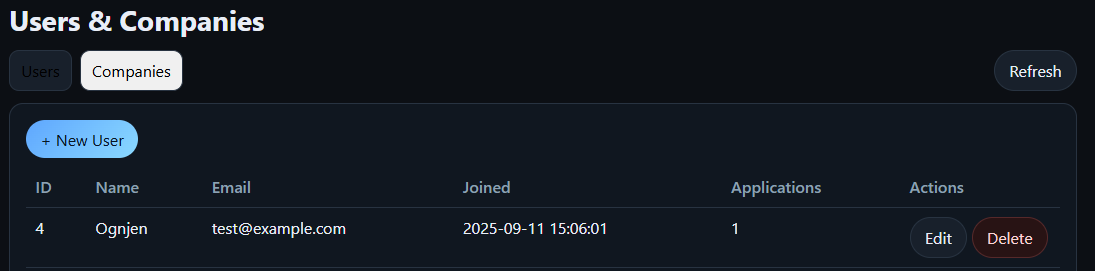
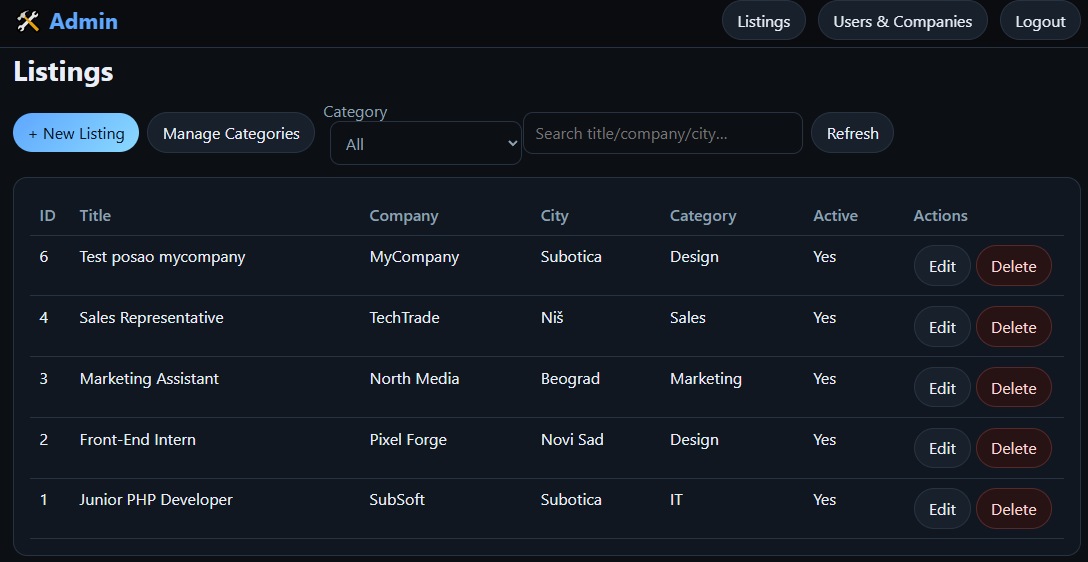


Klikom na „Contact“ dugme kompanija dobija prompt da upiše Subject i Messege, Tim putem mogu da obaveste korisnika da je primljen za posao ili da dogovore dalji način komuniciranja van sajta kako bi organizovali neki Interview. Korisniku preko email servisa stiže mail od strane „Job Portal“



Admin dashboard se pristupa pomoću zadatog maila i lozinke, u ovom slušaju [admin@example.com](mailto:admin@example.com) I change\_me\_123

Kada se pristupi admin dashboard-u administratori mogu da vide sve trenutne oglase, I da ih editiju (aktiviraju) ili brišu, mogu i da kreiraju i brišu kategorije za oglase. Takođe mogu da pristupe stranici users and companies, gde mogu da brišu i edituju podatke o korisnicima i kompanijama



# Korišćena literatura

**Link ka github repo projekta:**

**https://github.com/evasion123/joblisting**

<http://www.w3schools.com/>

[http://www.jquery.com](http://www.jquery.com/)

https://www.php.net/manual/en/book.session.php

https://www.php.net/manual/en/function.password-hash.php

https://developer.mozilla.org/en-US/docs/Web/API/Fetch\_API

https://developer.mozilla.org/en-US/docs/Web/API/FormData

https://github.com/PHPMailer/PHPMailer

(https://github.com/PHPMailer/PHPMailer/wiki/Troubleshooting )

https://www.virtualmin.com/documentation/web/website

Guideline projekat

<https://github.com/gshanbhag525/Job-Portal>

https://github.com/fulfilen/job-portal