

UNIVERSITATEA DE STAT DIN MOLDOVA

FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

DEPARTAMENTUL INFORMATICĂ

Dare de seamă la stagiul de practică

Tema: Dezvoltarea unei aplicații web pentru gestionarea rețetelor culinare

Autor: Bularga Vlad

Chișinău, 2025

Contents

1. Introducere: Descrierea Proiectului și Obiectivele Stagiului.....	3
2. Descrierea Problemei și Soluția Propusă	3
3. Arhitectura Aplicației și Tehnologii Utilizate	4
3.1. Structura Directorului Proiectului	4
3.2. Tehnologii Frontend	5
3.3. Tehnologii Backend	5
3.4. Baza de Date	6
4. Logica Implementării și Funcționalitățile Aplicației	6
4.1. Inițializarea Bazei de Date (`db.js`).....	6
4.2. Logica Serverului (`server.js`)	7
4.3. Interfața Utilizatorului (Fișiere HTML și `script.js`)	9
4.4. Stilul și Experiența Utilizatorului (`style.css`)	11
5. Funcționalitățile Cheie ale Aplicației și Interacțiunea cu Utilizatorul	13
5.1. Înregistrarea Utilizatorilor	13
5.2. Autentificarea (Login).....	13
5.3. Vizualizarea Rețetelor (Pagina Principală)	14
5.4. Vizualizarea Detaliilor unei Rețete	14
5.5. Adăugarea unei Rețete Noi	15
5.6. Logout (Deconectare)	16
6. Securitatea și Validarea Datelor	16
7. Concluzii și Perspective de Dezvoltare Viitoare	17

1. Introducere: Descrierea Proiectului și Obiectivele Stagiului

Prezentul raport detaliază activitatea desfășurată în cadrul stagiului de practică, având ca scop dezvoltarea unei aplicații web complete pentru gestionarea rețetelor culinare. Proiectul, denumit "My Recipes App", simulează un sistem unde utilizatorii pot înregistra, autentifica, vizualiza și adăuga propriile rețete, facilitând organizarea și partajarea acestora într-o manieră eficientă și estetică.

Obiectivele principale ale stagiului au inclus:

- * Consolidarea cunoștințelor de programare web (Frontend și Backend).
- * Familiarizarea cu lucrul într-un mediu Node.js și Express.js.
- * Implementarea unei baze de date SQLite pentru persistența datelor.
- * Asigurarea securității prin autentificare și autorizare bazată pe JWT.
- * Crearea unei interfețe de utilizator moderne și responsive, punând accent pe experiența utilizatorului (UX) și design (UI).
- * Aplicarea principiilor de modularitate și separare a responsabilităților în structura codului.

2. Descrierea Problemei și Soluția Propusă

În contextul digital actual, gestionarea rețetelor culinare personale sau partajarea acestora poate fi o provocare fără un sistem organizat. Utilizatorii se confruntă adesea cu dispersarea informațiilor, dificultatea de a găsi rețete specifice și lipsa unei platforme centralizate pentru stocarea și vizualizarea acestora.

Soluția propusă este o aplicație web interactivă care permite:

- * Crearea conturilor de utilizator pentru acces securizat.

- * Adăugarea de rețete noi cu titlu, descriere scurtă, ingrediente, instrucțiuni și imagini.
- * Vizualizarea unei liste complete de rețete disponibile.
- * Consultarea detaliilor individuale ale fiecărei rețete.
- * O interfață de utilizator intuitivă și un design modern, care îmbunătățește experiența de navigare și utilizare.

Această aplicație este utilă pentru oricine dorește să-și organizeze colecția de rețete digital, să le partajeze sau să descopere rețete noi într-un format accesibil și plăcut.

3. Arhitectura Aplicației și Tehnologii Utilizate

Aplicația "My Recipes App" este construită pe o arhitectură client-server (Frontend și Backend), utilizând tehnologii JavaScript extinse pentru ambele părți.

3.1. Structura Directorului Proiectului

Structura proiectului este modulară, facilitând organizarea și mentenanța codului:

```
my-recipes-app/  
├── node_modules/  
├── public/  
│   ├── uploads/           (Uploaded images are stored here)  
│   ├── style.css          (Main CSS for styling)  
│   ├── index.html         (Homepage)  
│   ├── recipe.html        (Individual recipe detail page)  
│   ├── register.html      (User registration page)  
│   ├── login.html         (User login page)  
│   ├── create-recipe.html (Page for adding new recipes)  
│   └── script.js          (Shared JavaScript for frontend interactions)  
├── package.json  
├── package-lock.json  
├── db.js                  (Database initialization logic)  
└── server.js              (Backend server logic and API endpoints)
```

3.2. Tehnologii Frontend

Partea de Frontend este implementată utilizând tehnologii web standard:

- * HTML5: Pentru structura și conținutul paginilor web.
- * CSS3: Pentru stilizarea și aspectul vizual al aplicației. Au fost utilizate tehnici moderne de design (flexbox, grid), variabile CSS și animații pentru o experiență utilizator fluentă și estetică.
- * JavaScript (Vanilla JS): Pentru interactivitatea pe partea client, gestionarea evenimentelor, trimiterea cererilor către API-ul backend și actualizarea dinamică a interfeței.

3.3. Tehnologii Backend

Partea de Backend este construită pe platforma Node.js:

- * Node.js: Mediu de execuție JavaScript.
- * Express.js: Cadrul web minim și flexibil pentru Node.js, utilizat pentru crearea API-urilor RESTful și gestionarea rutelor.
- * bcrypt: O bibliotecă pentru hash-uirea parolelor, asigurând stocarea securizată a acestora în baza de date.
- * jsonwebtoken (JWT): Utilizat pentru crearea și verificarea token-urilor de autentificare, implementând un sistem de autentificare stateless și securizat.
- * cors: Middleware Express pentru a permite cererile cross-origin, esențial pentru comunicarea între frontend și backend pe porturi diferite în timpul dezvoltării.
- * multer: Middleware pentru gestionarea upload-urilor de fișiere (imagini) pe server.
- * path, fs: Module Node.js pentru lucrul cu căi de fișiere și sistemul de fișiere (crearea folderului de uploads, ștergerea fișierelor).

3.4. Baza de Date

- * SQLite3: O bază de date relațională ușoară, serverless și bazată pe fișiere, ideală pentru proiecte de dimensiuni mici și medii.
- * ``sqlite3`` (modul Node.js): Driver pentru interacțiunea cu baza de date SQLite din Node.js.
- * ``db.js``: Fișier dedicat pentru inițializarea bazei de date, crearea tabelelor ``users`` și ``recipes``, și asigurarea persistenței datelor.

4. Logica Implementării și Funcționalitățile Aplicației

4.1. Inițializarea Bazei de Date (``db.js``)

Fișierul ``db.js`` conține logica pentru inițializarea bazei de date SQLite. Această modularizare asigură o separare clară a responsabilităților, permițând ca logica de definire a schemelor de date să fie independentă de logica serverului.

Detalii de implementare:

- * Utilizează ``sqlite3.verbose()`` pentru o interacțiune detaliată cu baza de date.
- * Definește calea către fișierul bazei de date (``recipes.db``).
- * Funcția ``initializeDatabase()`` creează baza de date dacă nu există și definește două tabele esențiale:
 - * ``users``: Pentru stocarea informațiilor utilizatorilor (``id``, ``username``, ``password``). ``username`` este unic, iar ``password`` este stocat sub formă de hash.
 - * ``recipes``: Pentru stocarea detaliilor rețetelor (``id``, ``title``, ``short_description``, ``image_url``, ``ingredients``, ``instructions``, ``user_id``, ``created_at``). Câmpul ``user_id`` este o cheie externă către tabela ``users``, asigurând legătura dintre rețetă și autorul său.
- * S-a adăugat o operațiune ``ALTER TABLE`` pentru a asigura existența coloanei ``image_url`` în tabela ``recipes``, gestionând cazurile în care baza de date ar fi fost creată într-o versiune anterioară fără această coloană.

```

7 function initializeDatabase() {
8   const db = new sqlite3.Database(DB_PATH, (err) => {
9     if (err) {
10       console.error('Error opening database:', err.message);
11     } else {
12       console.log('Connected to SQLite database.');

```

4.2. Logica Serverului (`server.js`)

Fișierul `server.js` este inima aplicației backend. Acesta configurează serverul Express, definește rutele API și gestionează logica de afaceri, inclusiv autentificarea, autorizarea și operațiunile CRUD (Create, Read, Update, Delete) pentru rețete.

Detalii de implementare:

- * Inițializare: Importă Express, bcrypt, JWT, multer, etc., și inițializează baza de date apelând `initializeDatabase()` din `db.js`.

- * Middleware:

- * `express.json()`: Pentru parsarea corpurilor de cereri JSON.

- * `cors()`: Pentru a gestiona politicile CORS.

- * `express.static()`: Pentru a servi fișiere statice din directorul `public`.

- * Upload de Imagini: Utilizează `multer` pentru a gestiona încărcarea imaginilor, configurând destinația (`public/uploads`) și numele fișierului. Se asigură crearea directorului `uploads` dacă acesta nu există.

- * Autentificare JWT (`authenticateToken`): O funcție middleware care verifică prezența și validitatea unui token JWT în antetul `Authorization`. Aceasta protejează rutele care necesită autentificare.

- * API Endpoints:

* `POST /api/register`: Înregistrează un utilizator nou, hash-uind parola cu `bcrypt`.
Gestionează cazurile de utilizator existent.

* `POST /api/login`: Autentifică un utilizator, compară parola furnizată cu hash-ul stocat și generează un token JWT valid.

* `GET /api/recipes`: Returnează o listă cu toate rețetele, incluzând numele autorului (printr-un JOIN cu tabela `users`).

* `GET /api/recipes/:id`: Returnează detaliile unei rețete specifice, inclusiv numele autorului.

* `POST /api/recipes`: Creează o rețetă nouă. Această rută este protejată de middleware-ul `authenticateToken` și gestionează încărcarea imaginilor. Datele rețetei și URL-ul imaginii sunt stocate în baza de date.

```
app.post('/api/register', async (req, res) => {
  const { username, password } = req.body;
  if (!username || !password) {
    return res.status(400).json({ message: 'Username and password are required.' });
  }

  try {
    const hashedPassword = await bcrypt.hash(password, 10);
    db.run('INSERT INTO users (username, password) VALUES (?, ?)', [username, hashedPassword], function (err) {
      if (err) {
        if (err.message.includes('UNIQUE constraint failed: users.username')) {
          return res.status(409).json({ message: 'User with this username already exists.' });
        }
        return res.status(500).json({ message: 'Error registering user.', error: err.message });
      }
      res.status(201).json({ message: 'User registered successfully!' });
    });
  } catch (error) {
    res.status(500).json({ message: 'Server error.', error: error.message });
  }
});

app.post('/api/login', async (req, res) => {
  const { username, password } = req.body;
  if (!username || !password) {
    return res.status(400).json({ message: 'Username and password are required.' });
  }
});
```



```

app.post('/api/recipes', authenticateToken, upload.single('image'), (req, res) => {
  const { title, short_description, ingredients, instructions } = req.body;
  const user_id = req.user.id;
  const image_url = req.file ? `/uploads/${req.file.filename}` : null;

  if (!title || !ingredients || !instructions) {
    if (req.file) {
      fs.unlink(req.file.path, (err) => {
        if (err) console.error('Error deleting file:', err);
      });
    }
    return res.status(400).json({ message: 'Title, ingredients, and instructions are required.' })
  }

  db.run(
    `INSERT INTO recipes (title, short_description, image_url, ingredients, instructions, user_id)
    VALUES (${title}, ${short_description}, ${image_url}, ${ingredients}, ${instructions}, ${user_id})`,
    function(err) {
      if (err) {
        if (req.file) {
          fs.unlink(req.file.path, (unlinkErr) => {
            if (unlinkErr) console.error('Error deleting file after DB error:', unlinkErr);
          });
        }
        return res.status(500).json({ message: 'Error creating recipe.', error: err.message })
      }
      res.status(201).json({ message: 'Recipe created successfully!', id: this.lastID, image_url: image_url })
    }
  )
})

```

4.3. Interfața Utilizatorului (Fișiere HTML și `script.js`)

Interfața cu utilizatorul este compusă din mai multe pagini HTML, iar logica interactivă este gestionată de un fișier JavaScript comun.

Detalii de implementare:

- * `index.html`: Pagina principală care afișează o grilă de rețete.
- * `recipe.html`: Pagina de detalii pentru o rețetă selectată.
- * `register.html`: Formular pentru înregistrarea utilizatorilor.
- * `login.html`: Formular pentru autentificarea utilizatorilor.
- * `create-recipe.html`: Formular pentru adăugarea unei rețete noi, incluzând un câmp pentru upload de imagine.
- * `script.js`: Fișier JavaScript comun care:
 - * Gestionează starea de autentificare a utilizatorului (token și username stocate în `localStorage`).

* Actualizează dinamic bara de navigare pentru a afișa opțiunile "Login/Register" sau "Welcome, [Username]!/Create Recipe/Logout".

* Funcția `updateAuthUI()` este apelată pe fiecare pagină pentru a reflecta starea curentă de autentificare.

* Pe `index.html`, preia rețetele de la API și le randează sub formă de carduri.

* Pe `recipe.html`, preia detaliile unei rețete specifice pe baza ID-ului din URL și le afișează.

* Fiecare pagină de formular (`register.html`, `login.html`, `create-recipe.html`) are propriul bloc de script (în aceeași structură) care gestionează trimiterea datelor către API-ul backend, afișarea mesajelor de succes/eroare și redirectionarea utilizatorilor după operațiuni reușite.

```
index.html X
RecipesApp > public > index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>My Recipes</title>
7      <link rel="stylesheet" href="style.css">
8  </head>
9  <body>
10     <header>
11         <h1>My Recipes</h1>
12         <nav>
13             <a href="index.html">Home</a>
14             <span id="auth-links">
15                 <a href="login.html">Login</a>
16                 <a href="register.html">Register</a>
17             </span>
18             <span id="user-info" style="display: none;">
19                 Welcome, <span id="username-display"></span>!
20                 <a href="create-recipe.html">Create Recipe</a>
21                 <button id="logout-button">Logout</button>
22             </span>
23         </nav>
24     </header>
25     <main id="recipes-container">
26         <p>Loading recipes...</p>
27     </main>
28     <footer>
29         <p>&copy; 2025 My Recipes</p>
30     </footer>
```

```

updateAuthUI();

if (recipesContainer) {
  fetch('http://localhost:3000/api/recipes')
    .then(response => response.json())
    .then(recipes => {
      recipesContainer.innerHTML = '';
      if (recipes.length === 0) {
        recipesContainer.innerHTML = '<p>No recipes yet. Be the first to add one!</p>';
        return;
      }
      recipes.forEach(recipe => {
        const recipeBlock = document.createElement('div');
        recipeBlock.classList.add('recipe-block');
        recipeBlock.innerHTML = `
          
          <div class="recipe-content">
            <h2><a href="recipe.html?id=${recipe.id}">${recipe.title}</a></h2>
            <p>${recipe.short_description || 'No description provided.'}</p>
            <p class="author">By: ${recipe.username}</p>
          </div>
        `;
        recipesContainer.appendChild(recipeBlock);
      });
    });
}

```

4.4. Stilul și Experiența Utilizatorului (`style.css`)

Fișierul `style.css` a fost refactorizat pentru a oferi un design modern, curat și atractiv. Accentul a fost pus pe o paletă de culori armonioasă, tipografie lizibilă și animații subtile care îmbunătățesc interacțiunea.

Detalii de implementare:

- * Variabile CSS (`:root`): Definirea unei palete de culori și a fonturilor pentru o gestionare ușoară și coerență vizuală. S-a optat pentru nuanțe de albastru/gri cu un accent portocaliu/ambră.
- * Tipografie: Fonturi `Open Sans` (pentru titluri) și `Roboto` (pentru textul principal) pentru o citire plăcută și un aspect modern.
- * Layout Responsive: Utilizarea `flexbox` și `CSS Grid` pentru a asigura că aplicația arată bine pe diferite dimensiuni de ecrane (desktop, tabletă, mobil) prin `@media` queries.
- * Design al Elementelor:
 - * Cardurile de rețete (`recipe-block`) au colțuri rotunjite, umbre subtile și o animație de transformare (`translateY`, `box-shadow`) la hover pentru a indica interactivitatea.

* Formularele beneficiază de un aspect curat, cu animații de `fadeIn` la încărcare, feedback vizual pentru input-uri la focus și butoane cu gradient și efecte la hover/active.

* Elementele de navigare și footer-ul sunt stilizate pentru a complementa designul general.

* Animații: Implementarea de animații CSS (ex. `fadeIn`, `slideDown`) pentru tranziții fluide și o experiență utilizator îmbunătățită.

```
100  main {
101    flex: 1;
102    padding: 40px 20px;
103    max-width: 1300px;
104    margin: 40px auto;
105    background-color: var(--white);
106    border-radius: var(--border-radius-md);
107    box-shadow: var(--shadow-light);
108    box-sizing: border-box;
109  }
110
111  #recipes-container {
112    display: grid;
113    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
114    gap: 30px;
115    padding: 10px;
116  }
117
118  .recipe-block {
119    background-color: var(--white);
120    border-radius: var(--border-radius-md);
121    overflow: hidden;
122    box-shadow: var(--shadow-light);
123    display: flex;
124    flex-direction: column;
125    text-align: center;
126    transition: transform 0.3s ease, box-shadow 0.3s ease;
127    border: 1px solid var(--border-color);
128  }
```

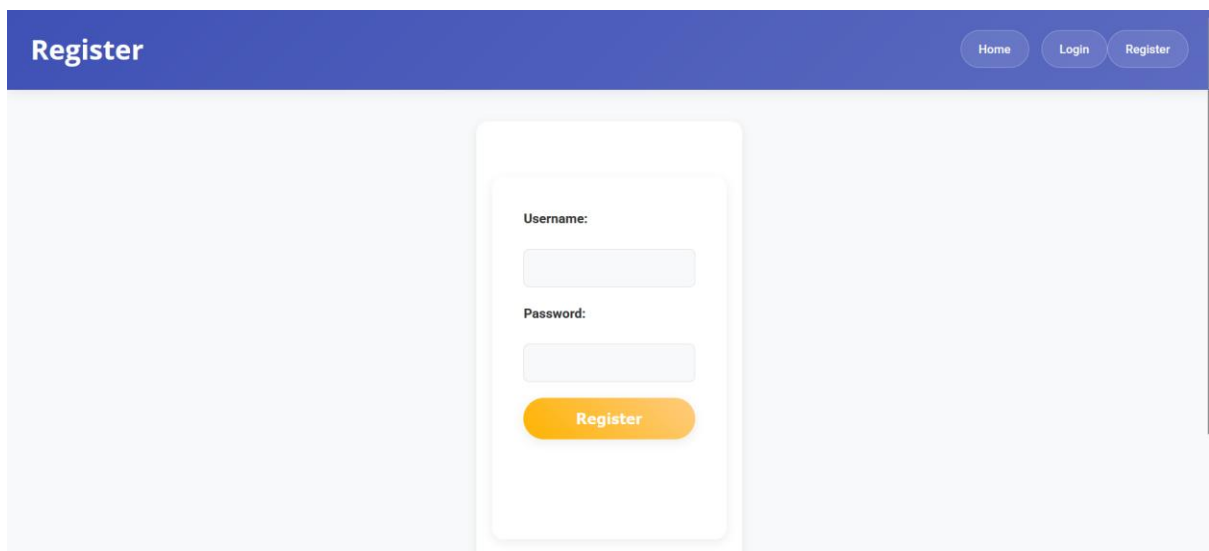
5. Funcționalitățile Cheie ale Aplicației și Interacțiunea cu Utilizatorul

5.1. Înregistrarea Utilizatorilor

Utilizatorii noi pot crea un cont furnizând un username și o parolă.

Interacțiune: Accesibil din bara de navigare via "Register". După completarea formularului, datele sunt trimise către `POST /api/register`. Parola este hash-uită înainte de stocare.

Mesajele de succes/eroare sunt afișate dinamic.

The image shows a web application interface for user registration. At the top, there is a dark blue header bar with the word "Register" in white on the left. On the right side of the header, there are three rounded buttons: "Home", "Login", and "Register", each with white text. Below the header, the main content area has a light blue background. In the center, there is a white rectangular card with a subtle shadow. Inside this card, there are two input fields: the first is labeled "Username:" and the second is labeled "Password:". Below these fields is a prominent orange button with the word "Register" in white text.

5.2. Autentificarea (Login)

Utilizatorii înregistrați se pot autentifica pentru a accesa funcționalități protejate.

Interacțiune: Accesibil din bara de navigare via "Login". După trimiterea username-ului și parolei către `POST /api/login`, serverul returnează un token JWT și username-ul utilizatorului, stocate local în browser (`localStorage`).

Login

HomeLoginRegister

Username:

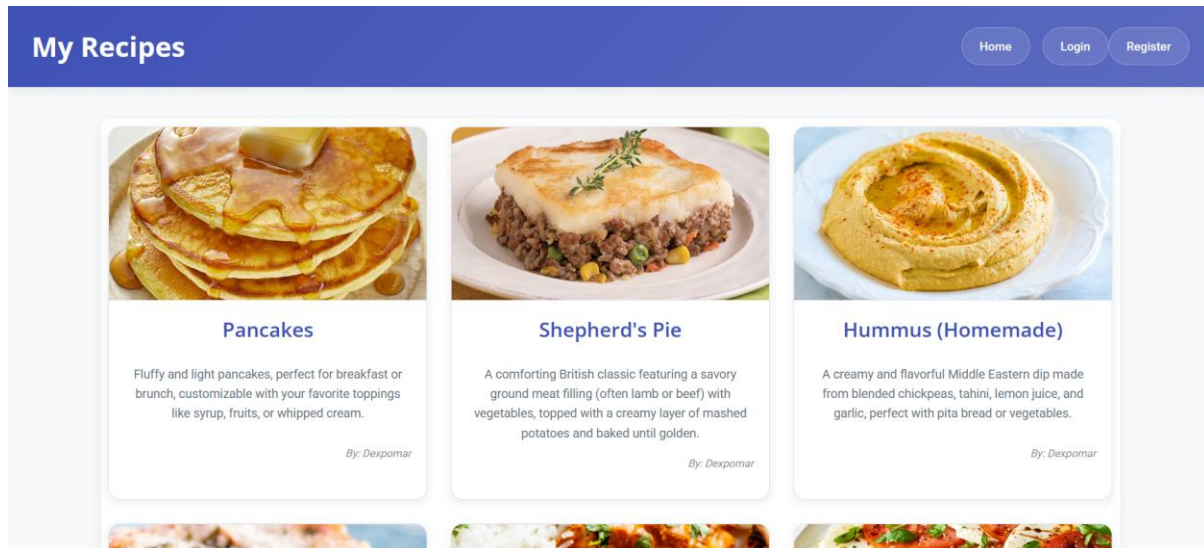
Password:

Login

5.3. Vizualizarea Rețetelor (Pagina Principală)

Pagina principală afișează o listă de carduri pentru rețete, cu titlu, descriere scurtă și imagine.

Interacțiune: La încărcarea paginii `index.html`, un request `GET /api/recipes` este trimis către server. Rețetele sunt apoi iterate și randate dinamic pe pagină.

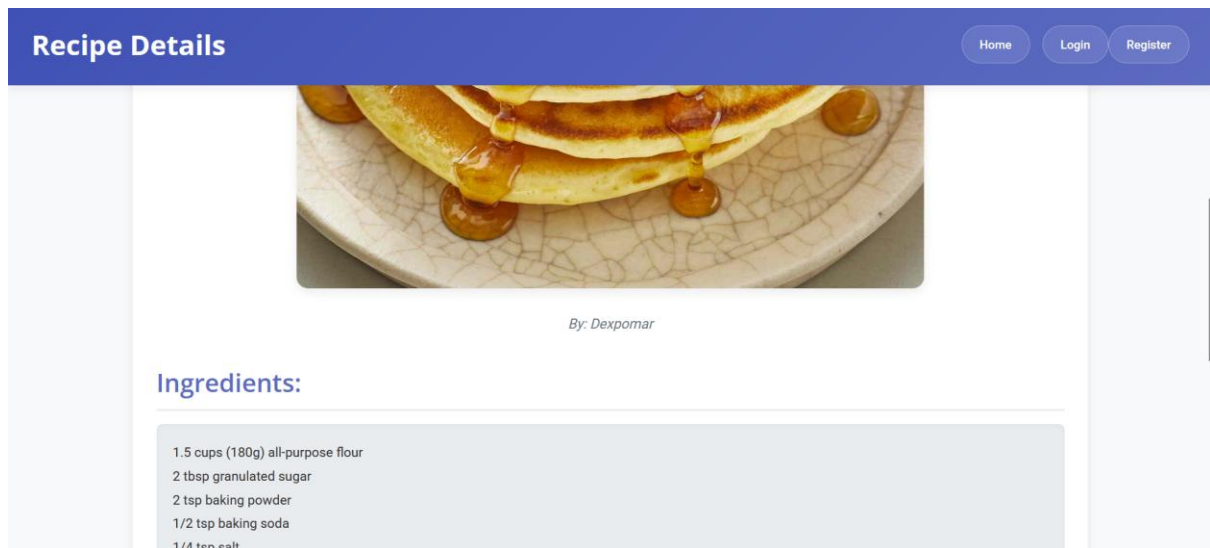


5.4. Vizualizarea Detaliilor unei Rețete

Fiecare card de rețetă de pe pagina principală este un link către o pagină dedicată cu detalii complete.

Interacțiune: Când se face click pe un card de rețetă, utilizatorul este redirecționat către `recipe.html?id=[recipe_id]`. Scriptul JS de pe această pagină preia `id`-ul din URL și face un

request `GET /api/recipes/:id` pentru a afișa ingredientele, instrucțiunile, imaginea și numele autorului.



5.5. Adăugarea unei Rețete Noi

Utilizatorii autentificați pot adăuga rețete noi.

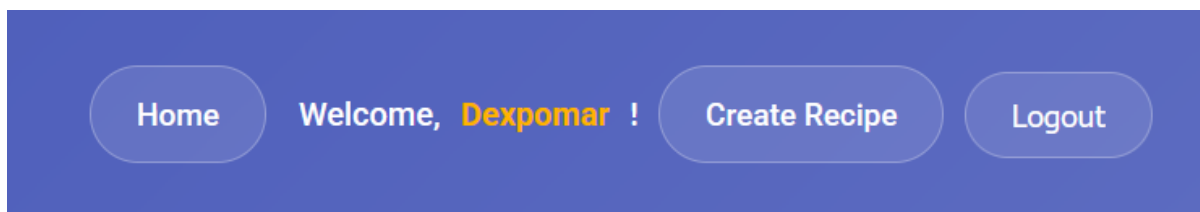
Interacțiune: Accesibil din bara de navigare via "Create Recipe" (doar dacă utilizatorul este logat). Formularul permite introducerea titlului, descrierii, ingredientelor, instrucțiunilor și încărcarea unei imagini. Datele sunt trimise ca `FormData` către `POST /api/recipes`, care este protejat de autentificarea JWT.

A screenshot of a web application's 'Create New Recipe' page. The page has a blue header with the title 'Create New Recipe' and four buttons: 'Home', 'Welcome, Dexpomar !', 'Create Recipe', and 'Logout'. Below the header is a form with the following fields: 'Recipe Title:' with a text input; 'Short Description:' with a text area; 'Image:' with a 'Browse...' button and the text 'No file selected.'; and 'Ingredients (each on a new line):' with a text area.

5.6. Logout (Deconectare)

Utilizatorii se pot deconecta pentru a încheia sesiunea.

Interacțiune: Un buton "Logout" este vizibil în bara de navigare când utilizatorul este autentificat. La click, token-ul JWT și username-ul sunt șterse din `localStorage`, iar UI-ul este actualizat.



6. Securitatea și Validarea Datelor

Pe parcursul dezvoltării, s-a pus accent pe aspectele de securitate și validare:

- * Hashing-ul Parolelor: Utilizarea `bcrypt` pentru a stoca parolele hash-uite, prevenind expunerea acestora în cazul unei breșe de securitate a bazei de date.
- * Autentificare JWT: Implementarea JSON Web Tokens pentru a gestiona sesiunile utilizatorilor într-un mod scalabil și sigur, fără a stoca starea sesiunii pe server.
- * Validarea Datelor de Intrare (Backend): Serverul verifică dacă câmpurile obligatorii sunt prezente (ex: username/password la înregistrare/login, title/ingredients/instructions la crearea rețetei).
- * Protecția Rutelor: Rutele critice (ex: `POST /api/recipes`) sunt protejate de middleware-ul `authenticateToken`, asigurând că doar utilizatorii autentificați pot efectua operațiuni de scriere.
- * Gestionarea erorilor de upload: Se șterge fișierul imagine încărcat dacă o eroare intervine la inserarea datelor în baza de date, prevenind fișiere orfane.

7. Concluzii și Perspective de Dezvoltare Viitoare

Aplicația "My Recipes App" a fost realizată cu succes, îndeplinind obiectivele stabilite ale stagiului de practică. S-a demonstrat capacitatea de a construi o aplicație web full-stack, utilizând Node.js, Express.js și SQLite, cu o interfață utilizator modernă și funcționalități esențiale pentru gestionarea rețetelor. Modularizarea codului și separarea responsabilităților au contribuit la o structură clară și ușor de înțeles.

Puncte forte ale implementării:

- * Arhitectură client-server clar definită.
- * Utilizarea JWT pentru autentificare securizată.
- * Design UI/UX îmbunătățit cu animații moderne.
- * Separarea logicii bazei de date de logica serverului.

Perspective de dezvoltare viitoare:

- * Funcționalitate CRUD pentru Rețete: Extinderea API-ului pentru a permite utilizatorilor autentificați să-și modifice (`PUT/PATCH /api/recipes/:id``) și să-și șteargă (`DELETE /api/recipes/:id``) propriile rețete.
- * Funcție de Căutare și Filtrare: Adăugarea unei funcționalități de căutare a rețetelor după titlu, ingrediente sau autor, precum și opțiuni de filtrare.
- * Funcționalitate de Favorit: Permitearea utilizatorilor să marcheze rețete ca favorite.
- * Paginare: Implementarea paginării pentru listele de rețete, pentru a gestiona volume mari de date.
- * Validare avansată pe Frontend: Adăugarea de validări suplimentare pe partea client pentru o experiență utilizator mai robustă.
- * Implementare de Recenzii/Comentarii: Posibilitatea utilizatorilor de a lăsa comentarii sau recenzii la rețete.
- * Deployment: Publicarea aplicației pe un serviciu de hosting (ex: Heroku, Vercel, Railway) pentru acces public.

Acest proiect a reprezentat o experiență valoroasă în aplicarea cunoștințelor teoretice în practică și a consolidat abilitățile de dezvoltare web full-stack.

<https://github.com/Dexpomar32/RecipesApp>