



UNIVERZITET
U NOVOM SADU
FAKULTET TEHNIČKIH
NAUKA U NOVOM SADU



Mastilović Radoslav

Reddit

PROJEKAT

Osnovne akademske studije

Novi Sad, 25.7.2024.

Sadržaj

1. Uvod	1
2. Opis korišćenih tehnologija i alata	2
3. Opis rešenja	4
3.4. <i>Enkripcija osetljivih podataka</i>	<i>4</i>
3.5. <i>Prikaz najaktivnijih postova</i>	<i>5</i>
3.6. <i>Like/dislike na komentare</i>	<i>6</i>
4. Opis praktičnog dela	8
4.1. <i>Funkcionalnost enkripcije</i>	<i>8</i>
4.2. <i>Funkcionalnost prikaza postova</i>	<i>9</i>
4.3. <i>Funkcionalnost like/dislike</i>	<i>13</i>
5. Zaključak	14
Literatura	15

1. Uvod

Ovaj dokument predstavlja primer projektne dokumentacije za nadogradnju postojeće aplikacije koja simulira rad "Reddit" platforme. Cilj ove nadogradnje je poboljšanje funkcionalnosti i sigurnosti aplikacije. Prvobitna aplikacija je omogućavala korisnicima: registraciju novog korisnika, logovanje postojećeg korisnika, izmenu korisničkog profila, pravljenje teme – novi post, pravljenje novog komentara od strane bilo kog korisnika, pretragu i sortiranje tema na osnovu naslova teme, brisanje teme ili ostavljenog komentara na neku temu, upvote – downvote teme, pretplata na temu.

Postojeća aplikacija je funkcionisala sa osnovnim funkcionalnostima, ali je nadogradnja bila neophodna kako bi se povećala interaktivnost, sigurnost i korisničko iskustvo. U okviru ove nadogradnje, dodate su sledeće funkcionalnosti:

1. Enkripcija osetljivih podataka - Implementirana je enkripcija za čuvanje osetljivih podataka, uključujući enkripciju lozinki koristeći algoritam za heširanje, tako da se uneti podaci ne mogu videti ni u bazi podataka niti bilo gde u aplikaciji.

2. Prikaz najaktivnijih postova - Na osnovu broja favorizovanja od strane korisnika, prikazuju se sortirane teme na početnoj stranici aplikacije kao posebna lista (od više ka manje favorizovanim). Sve ovo je povezano sa Azure Storage Explorer bazom podataka.

3. Like/dislike na komentare - Omogućava korisnicima da "lajkuju" ili "dislajkuju" komentare, čime se povećava interaktivnost i angažovanost korisnika. Ova funkcionalnost uključuje upis broja lajkova i dislajkova u Azure Storage Explorer bazu i njihovo čitanje.

2. Opis korišćenih tehnologija i alata

Ova sekcija pruža **detaljan pregled tehnologija i alata koji su korišćeni tokom realizacije projekta**. Cilj je objasniti izbor svakog alata i kako su doprineli celokupnom rešenju.

Projekat je realizovan u Visual Studio okruženju i sastoji se iz : AdminTools, Common, HealthMonitoringService, NotificationService, Reddit, RedditService i WebRole.

Objašnjenje korišćenih tehnologija i alata:

Microsoft Visual Studio [1] – je moćno integrisano razvojno okruženje (IDE) koje pruža sveobuhvatne alate za razvoj softvera. Omogućava programerima pisanje, uređivanje i debugovanje koda za različite programske jezike kao što su C#, C++, JavaScript, HTML, CSS i mnoge druge. Visual Studio je poznat po svojoj fleksibilnosti i mogućnosti proširenja putem dodataka, što ga čini prilagodljivim različitim potrebama razvoja. Podržava različite operativne sisteme uključujući Windows, macOS i Linux.

HTML (*HyperText Markup Language*) [2] – koristi se za strukturisanje sadržaja veb stranica. To je osnovni jezik za kreiranje veb stranica i veb aplikacija.

CSS (*Cascading Style Sheets*) [3] – koristi se za stilizaciju veb stranica, omogućavajući kontrolu nad izgledom i rasporedom elemenata na stranici, uključujući boje, fontove i raspored.

JavaScript [4] – koristi se za implementaciju interaktivnosti i komunikaciju sa serverom. Takođe se koristi za validaciju unosa podataka i upravljanje događajima na veb stranici.

Azure Cloud Service [5] – koristi se za hostovanje serverske strane aplikacije. Omogućava pouzdano i skalabilno upravljanje veb aplikacijama i servisima u cloud okruženju.

Azure Storage Service Explorer (Blob, Table, Queue) [6] – koristi se za skladištenje podataka aplikacije. Azure Blob Storage se koristi za nestrukturisane podatke, Azure Table Storage za strukturisane podatke, a Azure Queue Storage za upravljanje redovima poruka između komponenti aplikacije.

Razlog odabira tehnologija:

Microsoft Visual Studio [1] je odabran zbog svoje sveobuhvatne podrške za razvoj aplikacija, uključujući napredne alate za debugovanje i mogućnosti proširenja putem dodataka. HTML (*HyperText Markup Language*) [2], CSS (*Cascading Style Sheets*) [3] i JavaScript [4], su odabrani kao osnovne tehnologije za front-end razvoj zbog svoje standardizacije i široke primene u industriji.

Azure Cloud Service [5] je odabran zbog svoje pouzdanosti i integracije sa drugim Microsoftovim servisima, omogućavajući efikasno upravljanje i skaliranje aplikacije. Azure Storage Service Explorer (Blob, Table, Queue) [6] je odabran zbog svoje fleksibilnosti u upravljanju različitim tipovima podataka što je u aplikaciji ključno.

Projekti u okviru rešenja "Reddit" su organizovani na sledeći način:

1. AdminTools – Ovaj projekat sadrži alatke za administraciju sistema. Upravljanje administratorima, *real-time* komunikaciju izmedju klijenta i servera.

2. Common – Projekat koji sadrži zajedničke komponente i biblioteke koje koriste svi ostali projekti u rešenju. To uključuje osnovne klase, interfejsa i pomoćne funkcije koje se često koriste.

3. HealthMonitoringService – Servis za monitoring dostupnosti sistema. Ovaj servis redovno proverava stanje drugih servisa (RedditService i NotificationService) i beleži rezultate u bazu podataka. U slučaju problema, šalje obaveštenje administratorima.

4. NotificationService – Servis za slanje notifikacija putem e-maila. Kada korisnik ostavi komentar na neku temu, ovaj servis šalje obaveštenje svim pretplaćenim korisnicima na tu temu. Implementiran je kao zaseban Worker Role servis sa svim instancama.

5. Reddit – Glavni projekat koji implementira osnovne funkcionalnosti aplikacije, uključujući korisnički interfejs za kreiranje i upravljanje temama i komentarima, kao i interakciju sa korisnicima.

6. RedditService – Servisni sloj za sve korisničke zahteve. Ovaj servis obrađuje zahteve korisnika kao što su registracija, prijava, kreiranje novih tema i komentara, pretraga i sortiranje tema, kao i glasanje za teme. Takođe, komunicira sa NotificationService za slanje notifikacija.

7. WebRole – Web aplikacija koja pruža korisnički interfejs i prikaz informacija. Ova komponenta koristi tehnologije kao što su HTML, CSS i JavaScript za kreiranje interaktivnog i respozivnog korisničkog iskustva. WebRole takođe prikazuje podatke o dostupnosti sistema i najaktivnijim postovima.

3. Opis rešenja

U ovom delu dokumentacije detaljno su opisane ključne funkcionalnosti i implementacije rešenja. Objašnjava se kako je svaka funkcionalnost integrisana i koje su prednosti postignute implementacijom ovih rešenja.

3.1. Enkripcija osjetljivih podataka

Prilikom pokretanja aplikacije, prvo se pojavljuje stranica za prijavu (slika 3.1) koja dopušta ulazak na stranicu Home – Reddit (slika 3.3) samo u slučaju da je korisnik registrovan. Kada unesemo sve podatke na stranici za registraciju (slika 3.1), podaci se čuvaju u bazi podataka, a lozinka se hešuje pri upisu. Kada se registrovani korisnik vrati na stranicu za prijavu, unosiće potrebna polja, a program će iščitati hešovanu lozinku iz baze za korisnika sa tom mejl adresom, uporediti je sa unesenom lozinkom i ako je upoređivanje uspešno i lozinke se poklapaju, korisnik će biti prebačen na stranicu Home – Reddit. Na stranici za izmenu korisničkog profila (slika 3.1), nova lozinka će opet biti hešovana i upisana u bazu podataka ako korisnik promeni lozinku. Na slici 3.2 prikazana je baza gde je važno polje Password, u kojem se vidi hešovana lozinka.

Prijava

Email:

Lozinka:

Registracija

Ime:

Prezime:

Adresa:

Grad:

Drzava:

Broj telefona:

Email:

Lozinka:

Slika: 1647375907899.jpg



Profil

Ime:

Prezime:

Adresa:

Grad:

Drzava:

Broj telefona:

Email:

Lozinka:

Slika: No file chosen

Slika 3.1. Prikaz a) stranica "Prijava", b) stranica "Registracija", c) stranica "Profil"

The screenshot displays the Microsoft Azure Storage Explorer application. The main window shows a table named 'UserTableTemp' with the following data:

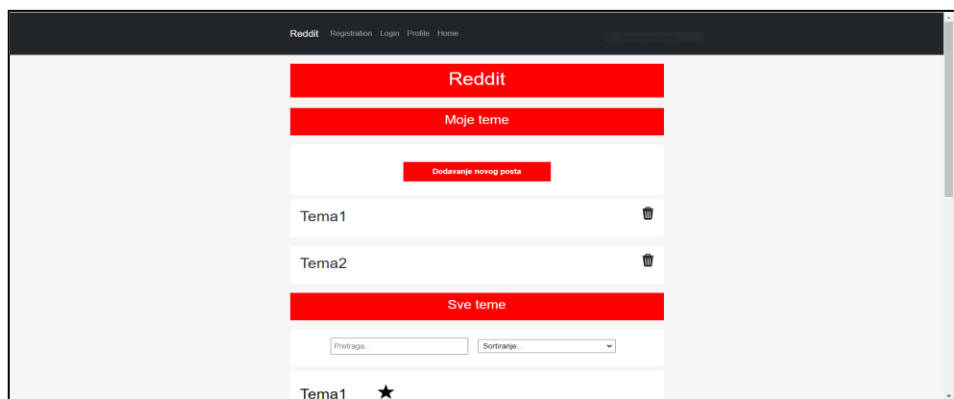
Country	PhoneNumber	Email	Password	Image
Serbia	060996915	radulov930@gmail.com	\$3a1135mLxns8b0uafL6At5C9...	/images/Profile/1647173007899.jpg
Serbia	066238147	vladimir123@gmail.com	\$3a1135mLxns8b0uafL6At5C9...	/images/Profile/images.jpg

The interface includes a sidebar on the left for navigation, a top toolbar with various actions (Query, Import, Export, etc.), and a bottom pane showing details for the selected table, including its display name, URL, type, and name.

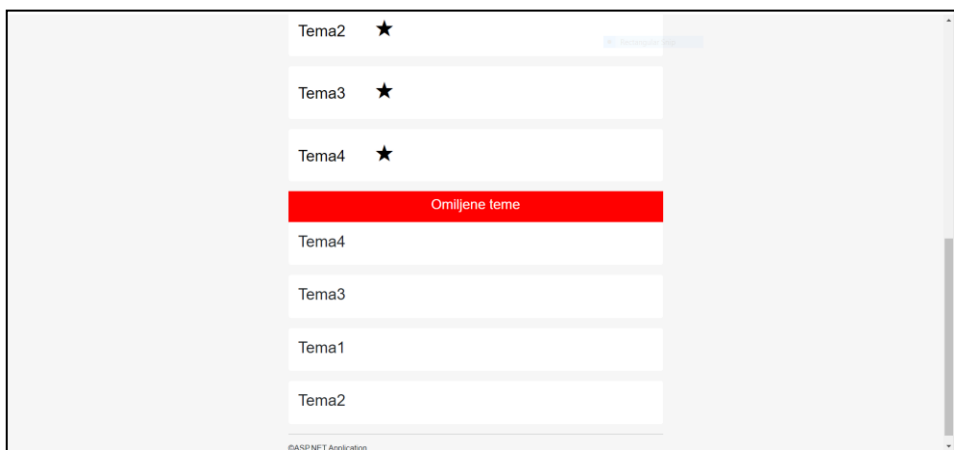
Slika 3.2. Prikaz hešovanih lozinki u tabeli "UserTableTemp" u bazi podataka

3.2. Prikaz najaktivnijih postova

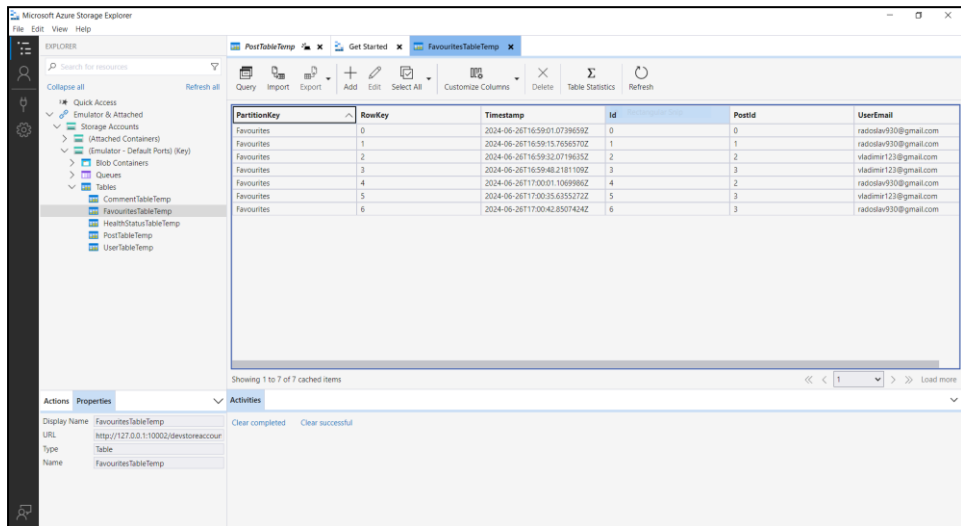
Kada se korisnici registruju i prijave na sistem, prelaze na Home – Reddit stranicu. Na slici 3.3 prikazane su funkcionalnosti i izgled, kao što su lista "Moje teme" koja prikazuje sve teme koje je pojedinačni korisnik dodao, i lista "Sve teme" gde se nalaze sve teme od svih korisnika na Reddit sistemu, sa mogućnošću sortiranja i pretrage postova prema nazivu, kao i brisanje tema. Pored toga, postoji mogućnost da se tema favorizuje kao omiljena od strane bilo kog korisnika. Kada neki korisnik favorizuje temu, ona se upisuje u posebnu tabelu u bazi podataka pod nazivom "FavouritesTableTemp", koja je prikazana na slici 3.5. Na osnovu te tabele i njenih polja potrebnih za sortiranje, podaci se iščitavaju u realnom vremenu i prikazuju u listi "Omiljene teme", sortirani od najviše do najmanje puta favorizovanih (slika 3.4).



Slika 3.3. Prikaz početne stranice sa ostalim funkcionalnostima



Slika 3.4. Prikaz liste "Omiljene teme" na početnoj stranici

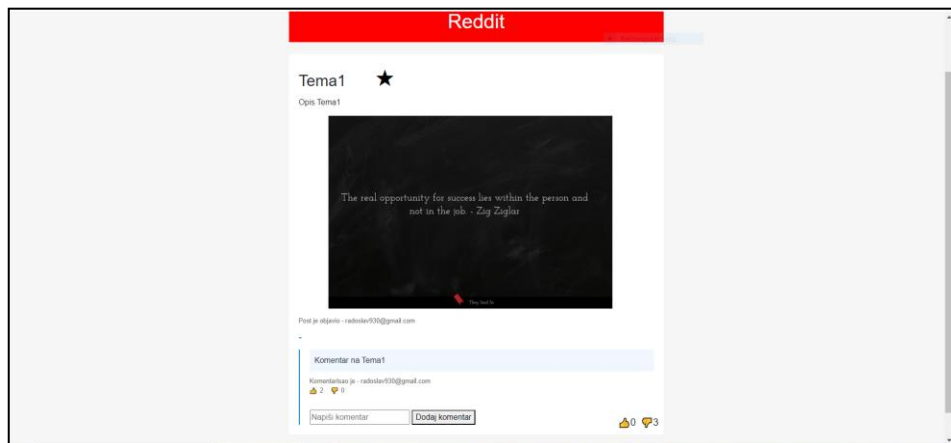


PartitionKey	RowKey	Timestamp	Id	PostId	UserEmail
Favourites	0	2024-06-26T16:59:01.0736959Z	0	0	radoslav930@gmail.com
Favourites	1	2024-06-26T16:59:13.7656870Z	1	1	radoslav930@gmail.com
Favourites	2	2024-06-26T16:59:32.0719635Z	2	2	vladimir123@gmail.com
Favourites	3	2024-06-26T16:59:48.2181109Z	3	3	vladimir123@gmail.com
Favourites	4	2024-06-26T17:00:01.1069986Z	4	2	radoslav930@gmail.com
Favourites	5	2024-06-26T17:00:25.6355272Z	5	3	vladimir123@gmail.com
Favourites	6	2024-06-26T17:00:42.8507424Z	6	3	radoslav930@gmail.com

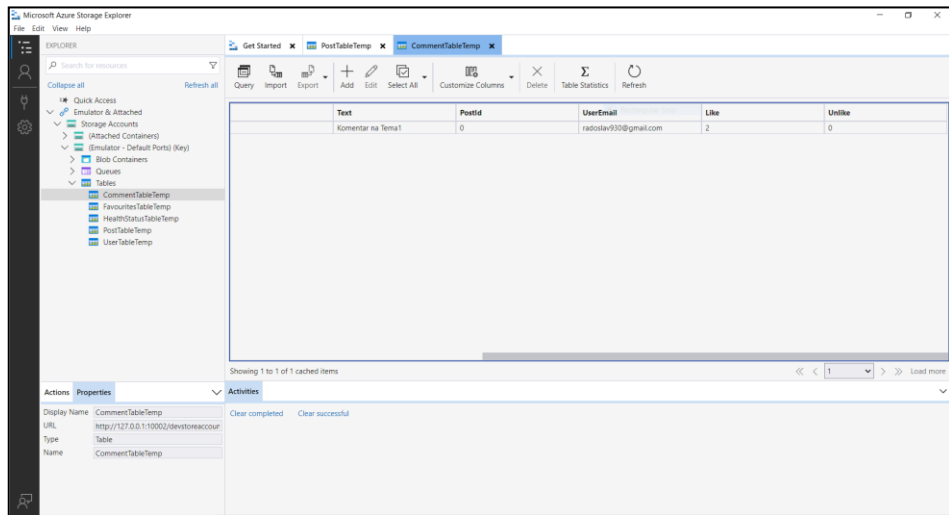
Slika 3.5. Prikaz tabele "FavouritesTableTemp" u bazi podataka

3.3. Like i Dislike na komentare

Na početnoj stranici imamo mogućnost da u listi "Sve teme" uđemo na neku od tema gdje će nam se prikazati izabrana tema (slika 3.6). Tu možemo da izvršimo like/dislike posta, a sve to je upisano u tabelu PostTableTemp (slika 3.8). Pored toga, imamo dodavanje i mogućnost like/dislike na postavljene komentare, što će se sve upisati u bazu podataka pod nazivom tabele CommentTableTemp (slika 3.7), gdje će biti prikazan broj tih like/dislike na komentaru, a to se iščitava i prikazuje na stranici.



Slika 3.6. Prikaz dodavanja like/unlike na komentar posta



Microsoft Azure Storage Explorer

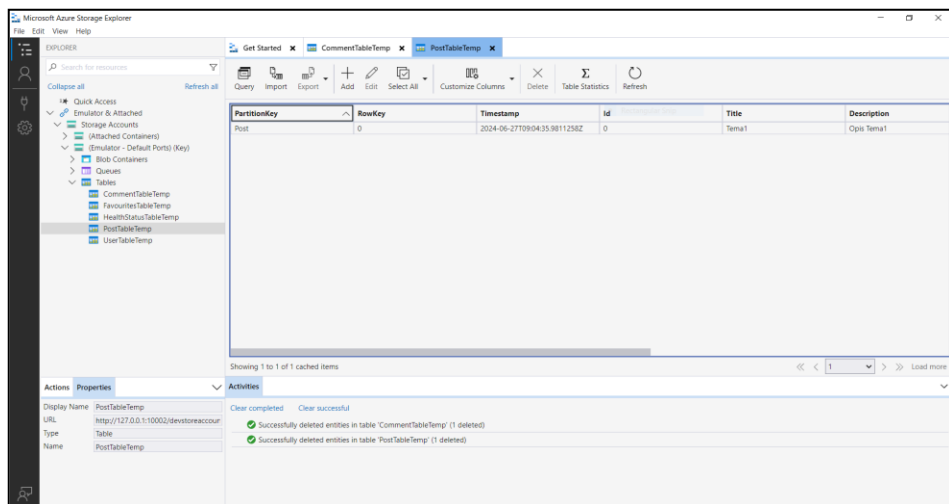
Explorer: CommentTableTemp

Text	PostId	UserEmail	Like	Unlike
Komentar na Tema1	0	radioslav930@gmail.com	2	0

Showing 1 to 1 of 1 cached items

Activities: Clear completed, Clear successful

Slika 3.7. Prikaz komentara u tabeli "CommentTableTemp"



Microsoft Azure Storage Explorer

Explorer: PostTableTemp

PartitionKey	RowKey	Timestamp	Id	Title	Description
Post	0	2024-06-27T09:04:35.9811258Z	0	Tema1	Opis Tema1

Showing 1 to 1 of 1 cached items

Activities: Clear completed, Clear successful, Successfully deleted entities in table 'CommentTableTemp' (1 deleted), Successfully deleted entities in table 'PostTableTemp' (1 deleted)

Slika 3.8. Prikaz posta u tabeli "PostTableTemp" kome pripada komentar

4. Opis praktičnog dela

Praktični deo projekta obuhvata implementaciju i testiranje svih funkcionalnosti. Ova sekcija pruža detaljan uvid u tehničke aspekte i konkretne primere kodiranja korišćenih metoda

4.1. Funkcionalnost enkripcije

Za potrebe enkripcije implementirana je klasa PasswordHasher (slika 4.1) u folderu Cryptography u projektu Common. U njoj se nalaze dve metode: HashPassword, koja prihvata lozinku kao ulazni parametar i vraća heširanu verziju te lozinke koristeći BCrypt.Net.BCrypt.HashPassword metodu, i VerifyPassword, koja prihvata lozinku i već heširanu lozinku kao ulazne parametre i vraća true ili false u zavisnosti od toga da li se lozinka podudara sa heširanom verzijom koristeći BCrypt.Net.BCrypt.Verify metodu.

```
namespace Common.Cryptography
{
    3 references
    public class PasswordHasher
    {
        2 references
        public static string HashPassword(string password)
        {
            return BCrypt.Net.BCrypt.HashPassword(password);
        }

        1 reference
        public static bool VerifyPassword(string password, string hashedPassword)
        {
            return BCrypt.Net.BCrypt.Verify(password, hashedPassword);
        }
    }
}
```

Slika 4.1. Prikaz klase "PasswordHasher" sa dve potrebne metode

Za potrebe enkripcije lozinke, metoda AddUser (slika 4.2) u klasi UserServiceProvider u servisu RedditService koristi PasswordHasher.HashPassword kako bi heširala lozinku pre nego što se kreirano novi korisnik koji će biti upisan u bazu podataka.

```
public void AddUser(UserData user)
{
    user.Password = PasswordHasher.HashPassword(user.Password);
    repository.Create(new UserData(user.Email) { FirstName = user.FirstName, LastName = user.LastName, Address = user.Address, City = user.City, Country = user.Country,
})
```

Slika 4.2. Prikaz metode za dodavanje korisnika u bazu podataka

Takodjer u metodi UpdateUser (slika 4.3.) koja se nalazi u klasi UserServiceProvider u servisu RedditService koristi PasswordHasher.HashPassword kako bi heširala lozinku pre nego što se izvrši ažuriranje u bazi podataka.

```
public void UpdateUser(string email, UserData user)
{
    user.Password = PasswordHasher.HashPassword(user.Password);
    repository.Update(email, new UserData(user.FirstName, user.LastName, user.Address, user.City, user.Country, user.PhoneNumber, user.Email, user.Password, user.Image))
}
```

Slika 4.3. Prikaz metode za ažuriranje korisnika u bazi podataka

Slika 4.4 prikazuje deo koda u kontroleru LoginController u folderu Controllers, koji se nalazi u servisu WebRole. Ovaj kod prikazuje pozivanje metode PasswordHasher.VerifyPassword i proverava da li je unesena lozinka za prijavu na sistem ista kao ona koja se nalazi u bazi podataka za tog korisnika. Ako jeste, korisnik se redirektuje na Home-Reddit stranicu, u suprotnom ostaje na istoj stranici.

```
if ((string.IsNullOrEmpty(email) && string.IsNullOrEmpty(password))
{
    UserData user = userService.GetUser(email);

    if (user != null && PasswordHasher.VerifyPassword(password, user.Password))
    {
        User loggedIn = new User(user.FirstName, user.LastName, user.Address, user.City, user.Country, user.PhoneNumber, user.Email, user.Password, user.Image);
        Session["LoggedInUser"] = loggedIn;

        // Logging received client data
        Debug.WriteLine("Email adresa: " + email);
        Debug.WriteLine("Lozinka: " + password);
        // Logging login status
        Debug.WriteLine("Uspesna prijava!");
        return RedirectToAction("Index", "Home");
    }
}
```

Slika 4.4. Prikaz provere lozinki pri logovanju na sistem

4.2. Funkcionalnost prikaza postova

Na slici 4.5. je prikazan kod koji je dodat u kontroler HomeController, uspostava veze sa servisom FavouriteService u kojem se nalaze sve potrebne metode. Prvo se kreira instanca ServiceConnector za interfejs IFavouritesService, a zatim se povezuje sa servisom koristeći Connect metodu sa zadatim URL-om. Nakon uspostavljanja veze, dobijeni proxy objekat favouritesService se koristi za dobavljanje svih i omiljenih postova iz tabele pozivanjem metode GetAllFavourites i GetAllPosts koji će biti potrebni zbog kasnijeg poredjenja kako bi se sortirala lista.

```
// Connect to the favourites service - DODATO
ServiceConnector<IFavouritesService> serviceConnectorFavourites = new ServiceConnector<IFavouritesService>();
serviceConnectorFavourites.Connect("net.tcp://localhost:10103/FavouritesService");
IFavouritesService favouritesService = serviceConnectorFavourites.GetProxy();

List<PostData> allPostsFromTable = postService.GetAllPosts();
List<CommentData> allCommentsFromTable = commentService.GetAllComments();
List<Favourites> allFavouritesFromTable = favouritesService.GetAllFavourites(); //DODATO
```

Slika 4.5. Prikaz povezivanja na "FavouriteService" i dobavljanje potrebnih podataka iz baze podataka

Slika 4.6 prikazuje metodu GetAllFavourites i metodu ReadAll. Metoda GetAllFavourites čita sve postove iz tabele FavouritesTableTemp pomoću metode ReadAll, kreira nove Favourites objekte i vraća ih kao listu. Metoda ReadAll vraća IQueryable<Favourites>, kreira i izvršava LINQ upit za čitanje svih stavki sa PartitionKey iz tabele u bazi. U slučaju greške, metoda baca izuzetak.

```
public List<Favourites> GetAllFavourites()
{
    return repository
        .ReadAll()
        .Select(fav => new Favourites
        {
            Id = fav.Id,
            PostId = fav.PostId,
            userEmail = fav.UserEmail,
        }).ToList();
}
```

a)

```
public IQueryable<Favourites> ReadAll()
{
    try
    {
        var results = from g in _table.CreateQuery<Favourites>()
            where g.PartitionKey == "Favourites"
            select g;
        return results;
    }
    catch (Exception ex)
    {
        throw new Exception($"{ex}");
    }
}
```

b)

Slika 4.6. Prikaz a) Metoda "GetAllFavourites", b) Metoda "ReadAll"

Funkcija `GetSortedFavouritesByPostId` se koristi za dobijanje sortiranih omiljenih postova (kasnije detaljnije o funkciji) a zatim se ti sortirani omiljeni postovi postavljaju u `AppContext.homePagePostLists`. `FavouritesPosts` kako bi se prikazali na pocetnoj stranici (slika 4.7).

```
private List<Post> GetSortedFavouritesByPostId(List<PostData> allPostsFromTable, List<Favourites> allFavouritesFromTable) //DODATA
{
    // Kreiranje mape za brojanje favorita
    var favouriteCounts = allFavouritesFromTable
        .GroupBy(f => f.PostId)
        .Select(group => new { PostId = group.Key, Count = group.Count() })
        .ToDictionary(g => g.PostId, g => g.Count);

    // Kreiranje liste postova sa brojem favorita
    List<Post> sortedFavorites = allPostsFromTable
        .Where(postData => favouriteCounts.ContainsKey(postData.Id))
        .Select(postData => new Post
        {
            Id = postData.Id,
            Title = postData.Title,
            Description = postData.Description,
            Image = postData.Image,
            UserEmail = postData.UserEmail,
            Like = postData.Like,
            Unlike = postData.Unlike,
            FavoriteCount = favouriteCounts[postData.Id]
        })
        .OrderByDescending(post => post.FavoriteCount)
        .ToList();

    return sortedFavorites;
}
```

Slika 4.7. Prikaz poziva funkcije za sortiranje liste i prosledjivanje za prikaz

Slika 4.8 prikazuje metodu `GetSortedFavoritesByPostId` koja kreira mapu brojeva favorita grupiranjem svih favorita po `PostId` i brojanjem koliko puta je svaki post označen kao omiljeni. Zatim filtrira postove iz `allPostsFromTable` koji se nalaze u mapi favorita, dodaje broj favorita kao atribut `FavoriteCount`, sortira postove opadajuće prema broju favorita i vraća sortiranu listu postova (`sortedFavorites`).

```
var sortedFavorites = GetSortedFavoritesByPostId(allPostsFromTable, allFavouritesFromTable); //DODATA FUNKCIJA

AppContext.homePagePostLists.AllPosts = allPosts;
AppContext.homePagePostLists.MyPosts = myPosts;
AppContext.homePagePostLists.FavoritesPosts = sortedFavorites; //DA SE PRIKAŽU - DODATO

return View(AppContext.homePagePostLists);
```

Slika 4.8. Prikaz metode za sortiranje liste za prikaz

Deo koda ispod prikazuje `index.cshtml` koji prvo proverava da li postoji omiljeni post u `Model.FavoritesPosts`. Ako nema nijednog omiljenog posta (`Count == 0`), prikazuje se prazan red. Ako postoje omiljeni postovi, koristi se `foreach` petlja za iteraciju kroz sve postove u `Model.FavoritesPosts` i prikazuje se naslov svakog posta unutar `div` elementa sa klasom `post`.

```
<!-- Postovi Favorites post -->
@if (Model.FavoritesPosts.Count == 0)
{
    <br>
}
else
{
    foreach (var post in Model.FavoritesPosts)
    {
        <div class="post">
            <h2>@post.Title</h2>
        </div>
    }
}
```

4.3. Funkcionalnost like/dislike

Za potrebe prikaza like/unlike u PostPage.cshtml je dodat deo koda HTML (*HyperText Markup Language*) [2], strukture (slika 4.10.) za prikazivanje komentara sa funkcionalnošću za lajkovanje i dislajkovanje. foreach petlja prolazi kroz sve komentare u Model.Comments gdje se nalaze podaci za komentare koji su proslijeđeni kroz PostPage kontroler. Svaki komentar se prikazuje unutar div elementa sa klasom comment-wrapper, koji sadrži tekst komentara, email korisnika koji je komentarisao, kao i dugmiće za lajkovanje i dislajkovanje. Dugmići prikazuju broj lajkova i dislajkova za svaki komentar koristeći comment.Like i comment.Unlike i imaju data-comment-id atribut za identifikaciju komentara.

```
<!-- Komentari -->
<div class="comments">
  @foreach (var comment in Model.Comments)
  {
    <div class="comment-wrapper">
      <div class="comment">@comment.Text</div>
      <div class="comment-meta">
        Komentarisao je - @comment.UserEmail
        <div class="like-unlike-container2">
          <div class="like-comment" data-comment-id="@comment.PostId">&#x1F44D;</div>
          <div class="like-count-comment" data-comment-id="@comment.PostId">@comment.Like</div>
          <div class="unlike-comment" data-comment-id="@comment.PostId">&#x1F44E;</div>
          <div class="unlike-count-comment" data-comment-id="@comment.PostId">@comment.Unlike</div>
        </div>
      </div>
    </div>
  }
</div>
<div style="margin-bottom:20px;"></div>
```

Slika 4.10. Prikaz podataka za komentare na stranici za dati post

Slika 4.11. je prikazuje JavaScript kod koji implementira funkcionalnost lajkovanja i dislajkovanja komentara. Kada se DOM potpuno učita (DOMContentLoaded), definišu se nizovi za skladištenje brojeva lajkova i dislajkova, kao i dugmića za lajkovanje i dislajkovanje. Dugmići likeCommentButtons i unlikeCommentButtons dobijaju event listener-e za klik događaje. Kada se klikne na dugme za lajkovanje, broj lajkova se povećava, a kada se klikne na dugme za dislajkovanje, broj dislajkova se povećava. Svaki klik šalje POST zahtev serveru sa ID-om komentara. Ako je zahtev uspešan, ažurira se prikaz sa novim brojem lajkova ili dislajkova, omogućavajući dinamičko ažuriranje broja lajkova i dislajkova za komentare u realnom vremenu

```

document.addEventListener("DOMContentLoaded", function () {
    const likeCountsComments = [];
    const unlikeCountsComments = [];
    const likeCommentButtons = document.querySelectorAll('.like-comment');
    const unlikeCommentButtons = document.querySelectorAll('.unlike-comment');
    const likeCountDisplays = document.querySelectorAll('.like-count-comment');
    const unlikeCountDisplays = document.querySelectorAll('.unlike-count-comment');
    likeCommentButtons.forEach((button, index) => {
        likeCountsComments[index] = parseInt(likeCountDisplays[index].textContent);

        button.addEventListener('click', () => {
            const commentId = button.getAttribute('data-comment-id');
            likeCountsComments[index]++;
            likeCountDisplays[index].textContent = likeCountsComments[index];

            fetch('/PostPage/LikeComment', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ commentId: commentId })
            }).then(response => response.json()).then(data => {
                if (data.success) {
                    likeCountDisplays[index].textContent = data.newLikeCount; // DODATO: Ažuriranje prikaza sa novim brojem lajkova za komentare
                }
            });
        });
    });

    unlikeCommentButtons.forEach((button, index) => {
        unlikeCountsComments[index] = parseInt(unlikeCountDisplays[index].textContent);
        button.addEventListener('click', () => {
            const commentId = button.getAttribute('data-comment-id');
            unlikeCountsComments[index]++;
            unlikeCountDisplays[index].textContent = unlikeCountsComments[index];

            fetch('/PostPage/UnlikeComment', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify({ commentId: commentId })
            }).then(response => response.json()).then(data => {
                if (data.success) {
                    unlikeCountDisplays[index].textContent = data.newUnlikeCount; // DODATO: Ažuriranje prikaza sa novim brojem dislajkova za komentare
                }
            });
        });
    });
});

```

Slika 4.11. Prikaz "JavaScript" koda za "like/dislike" na komentare

Na slici 4.12. prikazane su dve metode, LikeComment i UnlikeComment, koje omogućavaju komunikaciju sa JavaScript kodom (slika 4.11) i sa bazom podataka (slika 4.13. i slika 4.14) za lajkovanje i dislajkovanje komentara. Obe metode se povezuju sa CommentService putem ServiceConnector, dohvataju podatke o komentaru koristeći GetComment metodu na osnovu commentId, zatim povećavaju broj lajkova (Like) ili dislajkova (Unlike) za taj komentar, ažuriraju podatke o komentaru koristeći UpdateComment metodu i vraćaju JSON odgovor sa statusom uspeha i novim brojem lajkova ili dislajkova.

```

[HttpPost]
0 references
public ActionResult LikeComment(int commentId)
{
    ServiceConnector<ICommentService> serviceConnector = new ServiceConnector<ICommentService>();
    serviceConnector.Connect("net.tcp://localhost:10102/CommentService");
    ICommentService commentService = serviceConnector.GetProxy();
    CommentData commentData = commentService.GetComment(commentId);
    commentData.Like++;
    commentService.UpdateComment(commentData); // Promijenjeno: Korištenje metode UpdateComment
    return Json(new { success = true, newLikeCount = commentData.Like });
}

// POST: PostPage/UnlikeComment
[HttpPost]
0 references
public ActionResult UnlikeComment(int commentId)
{
    ServiceConnector<ICommentService> serviceConnector = new ServiceConnector<ICommentService>();
    serviceConnector.Connect("net.tcp://localhost:10102/CommentService");
    ICommentService commentService = serviceConnector.GetProxy();
    CommentData commentData = commentService.GetComment(commentId);
    commentData.Unlike++;
    commentService.UpdateComment(commentData); // Promijenjeno: Korištenje metode UpdateComment
    return Json(new { success = true, newUnlikeCount = commentData.Unlike });
}

```

Slika 4.12. Prikaz metoda za potrebe "LikeComment" i "UnlikeComment"

Slika 4.13. prikazuje metodu `AddComment` koja kreira novi komentar koristeći podatke iz objekta `CommentData` i koristi `repository.Create` metodu za dodavanje novog komentara u repozitorijum, uključujući ID, tekst, ID posta, email korisnika, broj lajkova i broj dislajkova. Metoda `GetComment` dohvata komentar na osnovu ID-a koristeći `repository.Read` metodu i vraća objekat `CommentData` sa podacima o komentaru. Metoda `UpdateComment` ažurira postojeći komentar koristeći podatke iz objekta `CommentData` i koristi `repository.Update` metodu za ažuriranje komentara u repozitorijumu. Ove metode omogućavaju dodavanje, dohvaćanje i ažuriranje komentara u bazi podataka, uključujući nove atribute za broj lajkova i dislajkova.

```
{
    private readonly static CommentRepository repository = new CommentRepository();

    2 references
    public void AddComment(CommentData comment)
    {
        repository.Create(new CommentData(comment.Text) { Id = comment.Id,
            Text = comment.Text, PostId = comment.PostId, UserEmail = comment.UserEmail,
            Like = comment.Like, Unlike = comment.Unlike });
    }

    3 references
    public CommentData GetComment(int id)
    {
        CommentData comment = repository.Read(id);
        return comment;
    }

    3 references
    public void UpdateComment(CommentData comment) // Dodato: Nova metoda za ažuriranje komentara
    {
        repository.Update(comment);
    }
}
```

Slika 4.13. Funkcije "AddComment", "GetComment", "UpdateComment" za komunikaciju sa bazom podataka

Slika 4.14. prikazuje metodu koja `Create` ubacuje novi komentar koristeći `TableOperation.Insert` i izvršava operaciju sa `_table.Execute`, baca izuzetak ako operacija nije uspešna. Metoda `Read` dohvata komentar po ID-u koristeći LINQ upit `ReadAll().Where(p => p.RowKey == id.ToString()).FirstOrDefault()`. Metoda `Update` ažurira postojeći komentar koristeći `TableOperation.Replace` i izvršava operaciju sa `_table.Execute`, baca izuzetak ako operacija nije uspešna. Sve metode koriste try-catch blokove za hvatanje i obradu izuzetaka.

```
public void Create(CommentData comment)
{
    try
    {
        TableOperation insertOperation = TableOperation.Insert(comment);
        TableResult result = _table.Execute(insertOperation);

        if (result.HttpStatusCode < 200 || result.HttpStatusCode >= 300)
        {
            throw new InvalidOperationException("Failed to insert user into Azure Table Storage.");
        }
    }
    catch (Exception ex)
    {
        throw new Exception($"{ex}");
    }
}

2 references
public CommentData Read(int id)
{
    return ReadAll().Where(p => p.RowKey == id.ToString()).FirstOrDefault();
}

2 references
public void Update(CommentData comment) // Dodato
{
    try
    {
        TableOperation updateOperation = TableOperation.Replace(comment);
        TableResult result = _table.Execute(updateOperation);

        if (result.HttpStatusCode < 200 || result.HttpStatusCode >= 300)
        {
            throw new InvalidOperationException("Neuspešno ažuriranje komentara u Azure Table Storage.");
        }
    }
    catch (Exception ex)
    {
        throw new Exception($"{ex}");
    }
}
```

Slika 4.14. Metode "Create", "Read" i "Update" potrebne za komunikaciju sa bazom podataka

5. Zaključak

Zaključak sumira glavne tačke projekta, uključujući poboljšanja u sigurnosti, interaktivnosti i integraciji sa Azure Storage Explorer. Takođe se pružaju predlozi za buduća unapređenja aplikacije.

1. Povećana sigurnost: Uvođenjem enkripcije osetljivih podataka, naročito lozinki, značajno je smanjena mogućnost kompromitacije korisničkih informacija, što povećava poverenje korisnika u aplikaciju.

2. Unapređena interaktivnost: Dodavanjem funkcionalnosti za lajkovanje i dislajkovanje komentara, kao i prikaz najaktivnijih postova, korisnicima je omogućeno da lakše prate i učestvuju u najpopularnijim diskusijama, čime se povećava angažovanost.

3. Efikasna integracija sa Azure Storage Explorer: Korišćenje Azure Storage Explorer-a za čuvanje i dohvaćanje podataka omogućava bolju skalabilnost i pouzdanost sistema, što je ključno za dalji rast aplikacije.

Predlozi za dalja usavršavanja:

1. Razvoj mobilne aplikacije: Kreiranje mobilne verzije aplikacije kako bi korisnici mogli pristupiti forumu sa svojih mobilnih uređaja, čime bi se dodatno povećala dostupnost i upotrebljivost aplikacije.

2. Napredne notifikacije: Implementacija sistema naprednih notifikacija koji bi korisnike obaveštavao o relevantnim aktivnostima u realnom vremenu, poput odgovora na njihove komentare ili novih objava u temama na koje su pretplaćeni.

3. Personalizacija sadržaja: Primena algoritama veštačke inteligencije za personalizaciju sadržaja i preporuke tema i komentara na osnovu korisničkih interesovanja i aktivnosti.

4. Proširenje analitičkih alata: Uvođenje naprednih analitičkih alata za praćenje aktivnosti korisnika, analizu popularnosti tema i komentara, te identifikaciju trendova u korišćenju aplikacije, što bi omogućilo donošenje informisanih odluka o daljim poboljšanjima.

Implementacija ovih predloga bi dodatno unapredila korisničko iskustvo i omogućila aplikaciji da ostane relevantna i konkurentna na tržištu, pružajući korisnicima sveobuhvatno i interaktivno okruženje za diskusije.

Literatura

[1] Microsoft Visual Studio. Pristup: 26.06.2024. Ključne reči: Visual Studio, razvojno okruženje, Microsoft. Dostupno na: [link](#).

[2] HTML (HyperText Markup Language). Pristup: 26.06.2024. Ključne reči: HTML, jezik za strukturisanje, veb stranice. Dostupno na: [link](#).

[3] CSS (Cascading Style Sheets). Pristup: 26.06.2024. Ključne reči: CSS, stilizacija, veb stranice. Dostupno na: [link](#).

[4] JavaScript. Pristup: 26.06.2024. Ključne reči: JavaScript, interaktivnost, programiranje. Dostupno na: [link](#).

[5] Azure Cloud Service. Pristup: 26.06.2024. Ključne reči: Azure, cloud, Microsoft. Dostupno na: [link](#).

[6] Azure Storage Service Explorer. Pristup: 26.06.2024. Ključne reči: Azure Storage, Explorer, Microsoft. Dostupno na: [link](#).

[7] ChatGPT. Pristup: 26.06.2024. Ključne reči: ChatGPT, OpenAI, AI pomoćnik. Dostupno na: [link](#).