



UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ INFORMATICĂ

LUCRARE DE LICENȚĂ

COORDONATOR:

Conf. Dr. Micotă Flavia Elena

ABSOLVENT:

Drăgoiu Iulia-Georgiana

TIMIȘOARA

2024

UNIVERSITATEA DE VEST DIN TIMIȘOARA
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ
PROGRAMUL DE STUDII DE LICENȚĂ INFORMATICĂ

UVTDorms

COORDONATOR:

Conf. Dr. Micotă Flavia Elena

ABSOLVENT:

Drăgoiu Iulia-Georgiana

TIMIȘOARA

2024

Abstract

The scope of this work is to create a useful platform for students in dormitories and their administrators. This web application will digitalize and automate many routine tasks in dormitories, such as creating appointments for the dormitory laundry, reporting problems, announcing events, managing students, and registering for dormitories.

Although the application presented in this paper is dedicated to the West University of Timișoara and its dormitories, the software can be extended and used by any other universities with a similar organizational structure.

The UVTDorms application is an easy-to-use platform for both students and dorm administrators, offering a simple, digitalized solution for students to create and manage laundry appointments, report problems quickly, and view upcoming events in the dormitory.

Dormitory administrators can manage the dormitory's washing machines and dryers (adding new ones if necessary), handle problem reports, and create new events. They can also manage the dormitory's rooms and students.

All these functionalities make it possible for any university with an organizational structure compatible with the project to use the software. Therefore, the application provides general utility and can help digitalize laundry management, event announcements, problem reporting, and student management in dormitories, offering numerous advantages to both students and dorm administrators.

Keywords: web application, java, angular, spring-boot, dorms, digitalization

Cuprins

Capitolul 1

Introducere

O bună organizare și o bună comunicare reprezintă doi factori foarte importanți în viața oricărei persoane, dar mai ales în viața oricărui student, în special în contextul vieții în cămin. Pentru a menține un mediu de trai plăcut, studenții trebuie să coabiteze cu alți studenți, iar acest lucru este posibil doar printr-o bună comunicare și organizare. Fiind un număr foarte mare de studenți, comunicarea și organizarea devin procese dificile, dacă se folosesc metode clasice. Căminele oferă diferite resurse studenților, precum mașinile de spălat și uscătoarele de rufe, spațiile comune (sală de lectură, bucătărie, băi), dar și întâlniri pe diferite tematici cu studenții locatari pentru o mai bună integrare și acomodare în cămin. O organizare eficientă și o comunicare bună în cadrul căminelor asigură utilizarea resurselor fără conflicte între locatari, dar și că informația ajunge în mod neeronat către toți locatarii.

Ca și studentă, dar și ca locatară în căminele din cadrul universității am avut nenumărate dificultăți și provocări în ceea ce privește viața în cămin. Atunci când mi-am ales tema lucrării de licență, mi-am dorit foarte mult ca aceasta să fie folositoare și să vină în ajutorul viitorilor studenți din cadrul universității, a studenților din cadrul proiectelor Erasmus, cât și a personalului administrativ. Astfel, aceasta să simplifice și să automatizeze multe din procesele din cadrul căminelor.

Aplicația își propune să ofere o modalitate de reducere a sarcinilor repetitive, centralizând gestionarea facilităților și resurselor comune. Aplicația include un sistem de autentificare și creare de conturi atât pentru studenți și administratorii de cămin. Studentul se poate înregistra furnizând anumite date personale, urmând ca administratorul, în urma notificării, să aprobe sau să respingă cererea de înscriere a studentului.

O funcționalitate de bază a aplicației o reprezintă programarea la mașina de spălat rufe, respectiv uscătoarele de rufe. Astfel, aceasta elimină procesul incomod de a face programarea pe foaie, reduce posibilele conflicte date de ștergerea intenționată a programărilor, risipa de hârtie. Administratorul căminului poate gestiona spălătoria prin modificarea statusului mașinilor în cazul în care apare vreo defecțiune, prin notificarea studenților pentru a se reprograma sau informarea lor că programarea a fost anulată. În plus, studentul poate deschide un tichet de reparații sau reclamații către administratorul căminului.

Suportul multilingv vine în ajutorul tuturor studenților, mai ales a studenților locatari din programul Erasmus, facilitând o bună integrare a acestora, dar și accesul facil și eficient la informații și resurse.

Administratorul căminului poate facilita comunicarea cu locatarii prin publicarea de anunțuri și evenimente, astfel încurajându-i pe locatari să participe la evenimente,

dar și îi ține la curent cu noile regulamente sau anunțuri legate de viața în cămin. De asemenea, acesta gestionează tichetele de reparații, astfel asigurându-se că fiecare tichet este redirecționat către personalul tehnic necesar, dar și monitorizează statusul fiecărei reparații. Pentru a înregistra camerele din cămin în aplicație, acesta poate adăuga, gestiona și vizualiza camerele și studenții cazați în camere, oferind un control al spațiilor din cămin.

Administratorul aplicației poate adăuga și gestiona crearea de conturi de administratori de cămine, dar și adăugarea de cămine și distribuirea administratorilor corespunzător căminelor.

1.1 Aplicații similare

Un rol important în dezvoltarea unei noi aplicații îl joacă aplicațiile similare. Compararea cu acestea reprezintă un pas de pornire, analizând punctele tari, punctele slabe sau cele lipsă, pentru a îmbunătății experiența utilizatorului, dar și pentru a decoperi potențialul unei inovări. După o căutare îndelungată, am aflat că în România, o singură aplicație abordează într-o anumită măsură această problemă, astfel aplicația UVTDorms reprezintă o inovație, nu doar la nivel local, dar și a țării. Pe lângă aplicația **me.utcj.app** [?], găsită la noi în țară, am mai găsit încă o aplicație similară: **TUSA** [?]. În subsecțiunile următoare, o să descriu aplicațiile, precum și punctele tari și slabe considerate de mine pentru fiecare aplicație în parte, iar la final, un tabel care să reprezinte o scurtă comparație a celor mai importante funcționalități.

1.1.1 me.utcj.app

Aplicația **me.utcj.app** este o aplicație mobilă destinată studenților locatari ai căminelor din cadrul Universității Tehnice din Cluj-Napoca. Aceasta oferă patru funcționalități de bază: vizualizarea notelor, vizualizarea meniului zilei de la cantină, programarea la mașina de spălat rufe și formularul de reparații. Studenții se conectează cu contul instituțional, astfel doar studenții universității pot avea posibilitatea de a beneficia de folosirea resurselor. Anterior aceștia au fost adăugați în sistem, fiind atribuiți căminului și camerei corespunzătoare. Aceștia pot să facă programare la mașina de spălat consultând un calendar unde pot vedea programările altor locatari și intervalele libere. Programarea se face pe baza numărului camerei și al numelui. De asemenea, studentul poate să deschidă un tichet de reparații, să vadă statusul acestuia sau tichetele anterioare deschise.

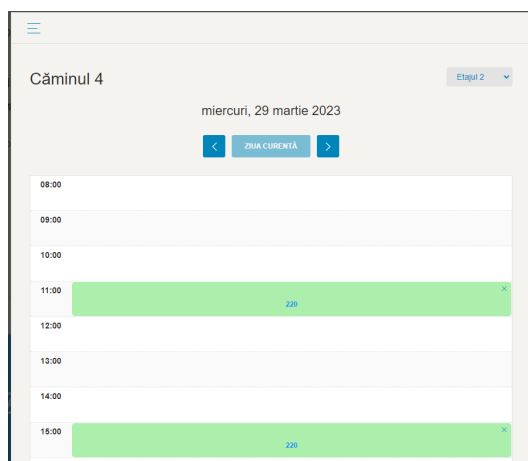


Figura 1.1: Creare programare la mașina de spălat

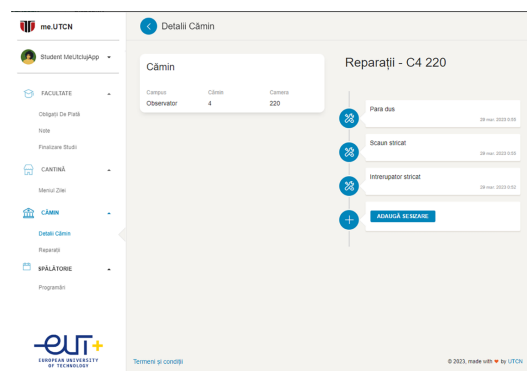


Figura 1.2: Istoric tichete reparații

1.1.2 TUSA

TUSA (Tasmanian University Student Association)¹ este o aplicație destinată studenților din cadrul Universității din Tasmania oferind sprijin pentru integrare, dar și sprijin și informații legate de resurse și servicii pentru o cât mai bună experiență studentescă. Aplicația oferă informații studenților sau viitorilor studenți despre posibilele oportunități, evenimente, cluburi sau voluntariate la care pot lua parte, cum se pot înscrie sau cum pot chiar ei să creeze un astfel de club sau societate. De asemenea, aplicația ajută studenții în ceea ce privește informarea despre beneficiile precum suport academic, consultanță financiară, alimentație sau locuințe. Studenții pot să își facă programare la mașina de spălat rufe conform orarului și disponibilității. Aceștia aleg ora și data disponibilă și își introduc numele, adresa de email și un număr de telefon. Accesul la programare nu este restricționat, deci și persoanele din afara universității pot face programare.

În plus, studenții sunt încurajați să ofere feedback despre experiența lor ca studenți, având o secțiune destinată pentru acest lucru. Pentru orice întrebare sau problemă, aceștia pot completa un formular în care să detalieze nelămurirea avută.

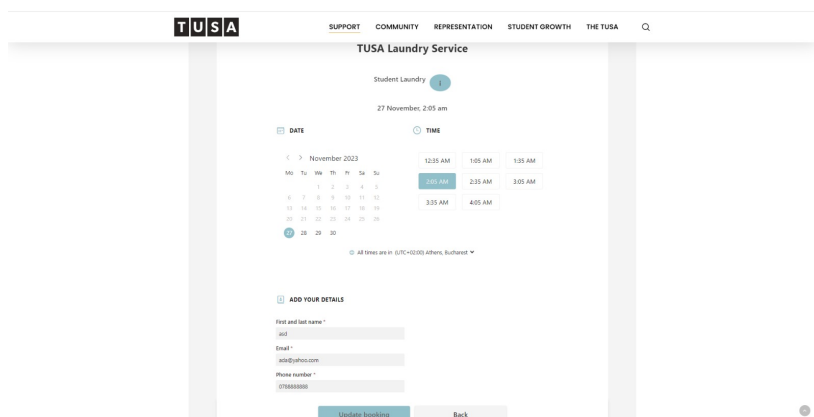


Figura 1.3: Creare programare la mașina de spălat rufe

¹<https://www.tusa.org.au/student-advocate-appointment/>

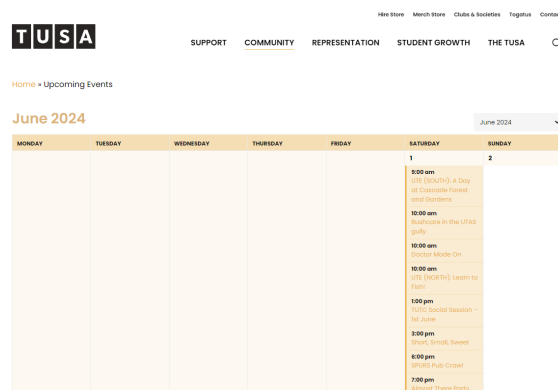


Figura 1.4: Vizualizare evenimente

1.2 Comparație aplicații asemănătoare

Prin tabelul de mai jos, am exemplificat o scurtă comparație între cele 2 aplicații similare și UVTDorms, subliniind astfel atât punctele tari cât și punctele slabe ale fiecăreia.

	me.utcj.app	TUSA	UVTDorms
Evenimente/anunțuri	Nu	Da	Da
Feedback	Nu	Da	Nu
Formular reparații	Da	Nu	Da
Validare cont	Da	Nu	Da
Programare limitată	Da	Nu	Da

Tabela 1.1: Comparație între aplicații

1.3 Structură lucrare

Lucrarea este structurată în cinci capitole și mai multe subsecțiuni care urmăresc să prezinte aplicația UVTDorms cu toate detaliile relevante. Primul capitol este destinat prezentării problemei, motivației și a obiectivelor, dar și o analiză și o comparație cu soluțiile deja existente. În următorul capitol, este prezentată arhitectura aplicației, exemplificând și detaliind prin diagrame corespunzătoare precum: diagrama arhitecturii aplicației, diagramele cazurilor de utilizare pentru fiecare tip de utilizator, diagrama structurii unităților de execuție, dar și diagrama bazei de date. Al treilea capitol cuprinde mai multe subsecțiuni în care sunt prezentate detalii de implementare a aplicației, atât în ceea ce privește back-end-ul, front-end-ul, configurarea bazei de date, dar și detalii despre securitate, serviciul de email și internaționalizarea aplicației. Capitolul patru este destinat utilizatorului, reprezentând un ghid de utilizare care îmbunătățește experiența utilizatorului și ajută la o mai bună înțelegere. Ultimul capitol, sumarizează detaliile prezentate în lucrare, dar și menționează posibilele direcții viitoare.

Capitolul 2

Arhitectura aplicației

Pentru o funcționare optimă și eficientă a aplicației, arhitectura acesteia a fost gândită într-un mod care permite modularizarea componentelor principale și care prevede cele mai bune metode de intercomunicare. Arhitectura aplicației privită de la o distanță care permite înțelegerea structurării principalelor componente ale aplicației poate fi reflectată prin următoarea diagramă, așa cum poate fi observată din Figura 2.1.

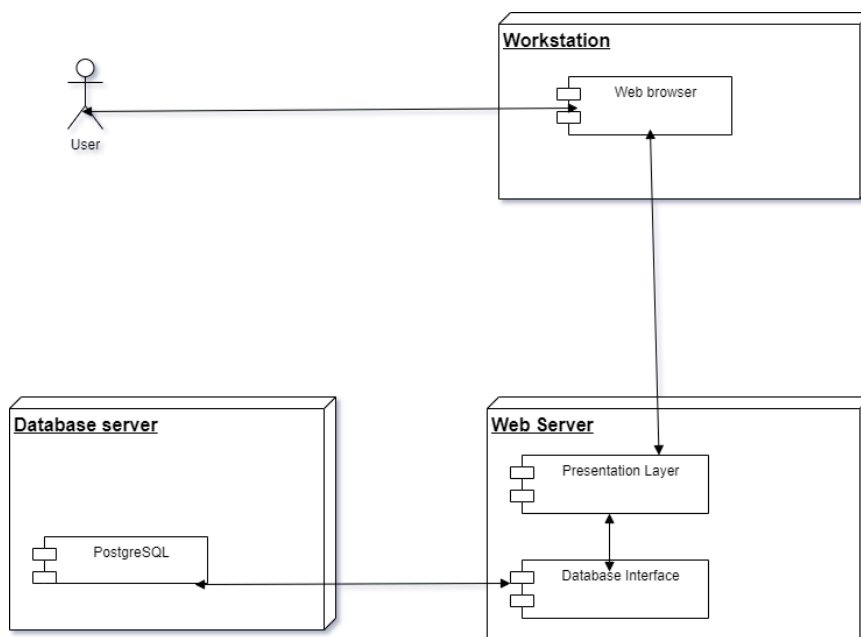


Figura 2.1: Diagrama arhitecturii aplicației

Așa cum reiese și din figura??, utilizatorul aplicației poate interacționa cu aceasta folosind un browser web. De asemenea, prin utilizarea infrastructurii oferite de browserele web, aplicația devine accesibilă utilizatorilor și prin utilizarea telefoanelor mobile.

Browserul web este modulul care reprezintă legătura directă dintre utilizator și componenta logică și funcțională a aplicației, serverul web. Acesta oferă serviciile principale ale aplicației accesibile prin stratul de prezentare, accesat direct de browserul web prin protocolul HTTPS.

HTTPS (Hyper Text Transfer Protocol/Secure) este un protocol folosit de rețele de calculatoare. Acesta criptează datele transmise de-a lungul rețelei de la un punct

la altul. Este predecesorul protocolului HTTP. Acesta oferă în plus securitate care se bazează pe certificatele bazate pe criptografia digitală care permite urmărirea autenticității utilizatorilor.

În momentul declanșării unor activități de prelucrare a datelor sau a unei cereri de date, browserul web accesează unul dintre end-point-urile stratului de prezentare a serverului web, care aplică logica funcționalității sale și returnează datele solicitate sau realizează modificările cerute.

Serverul web este responsabil doar pentru prelucrarea și manipularea datelor, dar nu și pentru stocarea acestora. Pentru o funcționare cât mai eficientă și optimă și pentru timp scurt de răspuns, stocarea datelor se întâmplă într-un modul separat al arhitecturii, pe serverul bazei de date, care este conectat în mod direct la serverul web prin interfața bazei de date. Astfel, de fiecare dată când serverul web trebuie să manipuleze datele aplicației, acesta le accesează din componenta separată.

Prin urmare, arhitectura aplicației este alcătuită din trei module principale, structurate într-o ierarhie de comunicare eficientă.

2.1 Cazuri de utilizare

Utilizatorii aplicației sunt studenții, administratorii de cămine și administratorul aplicației. Unul din cazurile de utilizare comune tuturor tipurilor de utilizatori este autentificarea care reprezintă o funcționalitate importantă prin care utilizatorii își accesează conturile. Alături de autentificare, recuperarea parolei, schimbarea acesteia, schimbarea numărului de telefon reprezintă de asemenea, cazuri de utilizare comune. Recuperarea parolei reprezintă o măsură de siguranță în cazul în care utilizatorul își pierde parola, ceea ce ar însemna și imposibilitatea de a accesa contul. Prin această funcționalitate, utilizatorii au posibilitatea de a-și salva contul și de a-și seta o parolă nouă. De asemenea, schimbarea parolei este posibilă din meniul de setări a tuturor tipurilor de utilizatori ca o formă de siguranță.

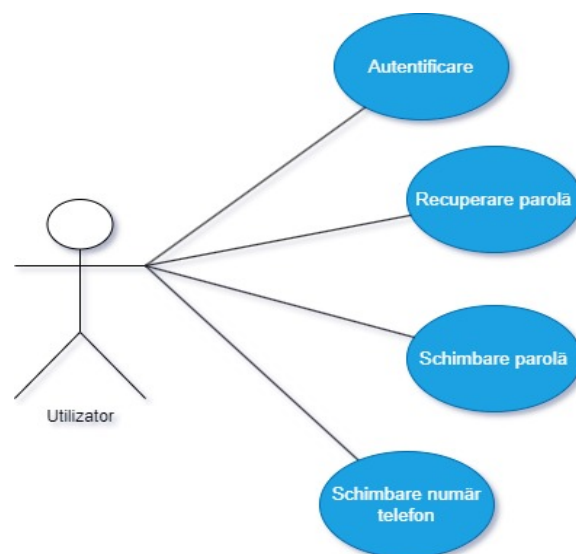


Figura 2.2: Diagrama cazurilor de utilizare comune

Cazurile de utilizare sunt reprezentate și detaliate pentru fiecare tip de utilizator

în subsecțiunile următoare.

2.1.1 Cazuri de utilizare pentru student

Principalul beneficiar al aplicației web este studentul locatar în căminul studențesc. Odată cazat în cămin, acesta poate să își creeze un cont și să solicite administratorului căminului din care face parte, validarea contului de student asociat cu căminul respectiv. După crearea cererii, studentul primește un mail de confirmare cu privire la realizarea cererii de înscriere și o parolă temporară pentru cont, pe care o poate schimba ulterior. Până la acceptarea cererii de înscriere, poate să intre în aplicație cu email-ul cu care și-a creat contul și parola primită, dar nu are acces la funcționalitățile oferite, rolul acestuia fiind de student inactiv.

Una dintre principalele funcționalități dedicată studenților este crearea programărilor la spălătoria căminului. Aceștia pot beneficia de o programare săptămânală, care este formată din două ore la mașina de spălat rufe, urmată de două ore la uscătorul de rufe. În cazul în care studentul dorește anularea programării, acesta poate anula programarea și, în limita disponibilităților, să realizeze o altă programare. Dacă pe perioada șederii în căminul studențesc apar diferite probleme tehnice care necesită intervenție din partea personalului, aceștia pot crea un tichet pentru reparații.

De asemenea, acesta poate vizualiza evenimentele adăugate de către administratorul căminului din care face parte, iar dacă acestea sunt deschise participării studenților, să își exprime dorința de participare.

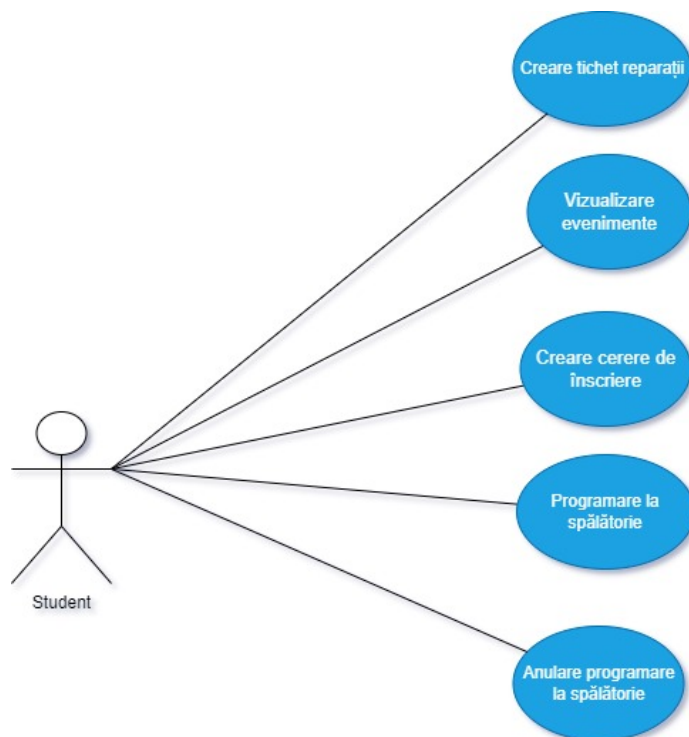


Figura 2.3: Diagrama cazurilor de utilizare specifice studentului

2.1.2 Cazuri de utilizare pentru administratorul de cămin

Administratorul de cămin este responsabil pentru administrarea unui cămin și al studenților cazați în căminul respectiv.

Acest rol prevede manipularea listei studenților în perioada de înscriere, adică acceptarea sau respingerea studenților care au un loc într-un cămin, sau dezactivarea conturilor studenților asociate căminului, dacă aceștia părăsesc sau schimbă căminul. Totodată, acesta poate adăuga camere în cămin și poate vizualiza informații legate de persoanele cazate în aceea cameră, având un rol important în gestionarea spațiului din cămin.

De asemenea, printre responsabilitățile administratorului se află și organizarea spălătoriei din cămin, adică administrarea mașinilor de spălat și a uscătoarelor de rufe (adăugarea, editarea și schimbarea statusului mașinilor dacă apar defecțiuni).

Gestionarea eficientă a defecțiunilor raportate de către locatari joacă un rol important în ceea ce privește confortul și prevenirea problemelor și defecțiunilor apărute pe termen lung, astfel, administratorul căminului este cel care redirecționează problemele apărute personalului, dar și monitorizează statusul rezolvării acestora.

Pe lângă cele descrise mai sus, administrarea evenimentelor ce au loc în cadrul căminului, se află tot pe lista sarcinilor administratorului căminului, care le poate crea, edita și șterge.

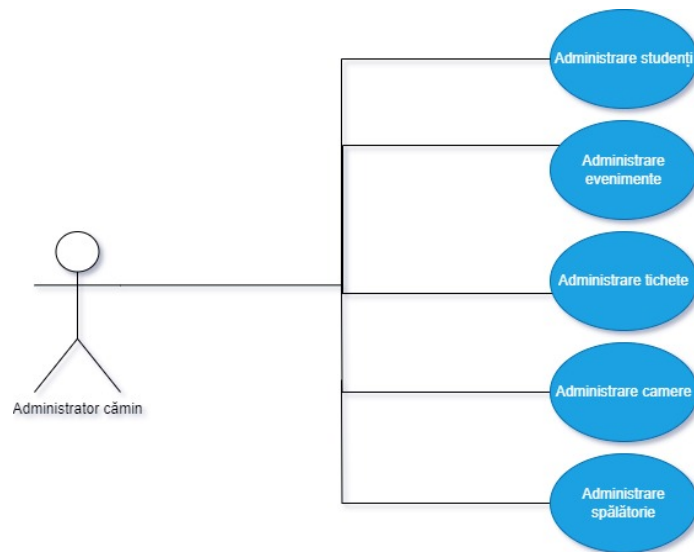


Figura 2.4: Diagrama cazurilor de utilizare specifice administratorului căminului

2.1.3 Cazuri de utilizare pentru administratorul aplicației

Administrarea aplicației la nivel înalt este o activitate crucială pentru buna ei funcționare. Acest rol prevede atât responsabilitățile legate de căminele universității care folosesc aplicația, cât și cele legate de administratorii acestor cămine.

Astfel, unul dintre principalele cazuri de utilizare ale administratorilor de aplicație este adăugarea de cămin, prin care sistemul căminelor universității poate fi extins.

Căminelor existente în sistem, administratorul aplicației le poate asocia câte un administrator. Totodată, și asocierea și ștergerea administratorilor se realizează de

administratorul aplicației. În acest caz, contul administratorului de cămin nu este șters, ci doar dezactivat de administratorul aplicației.

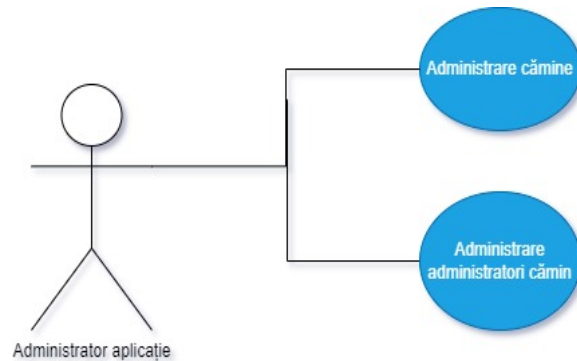


Figura 2.5: Diagrama cazurilor de utilizare specifice administratorului de aplicație

2.2 Structura unităților de execuție

Aplicația este structurată pe mai multe straturi, care la rândul lor sunt modularizate pentru o funcționare logică și eficientă.

Primul modul, cel al clientului, este alcătuit din trei componente ale căror cooperare oferă o experiență plăcută utilizatorului, fiind modulul care interacționează în mod direct cu acesta. Prima componentă, HTML GUI constituie interfața aplicației cu care ia contact utilizatorul declanșând acțiunile care reprezintă funcționalitățile aplicației, prin elemente de HTML precum butoane. Aceste acțiuni declanșate de componenta HTML GUI sunt reprezentate de componenta GUI function. Cea de-a doua componentă (GUI function) este responsabilă pentru activitățile logice ale primului modul și declanșează inițierea legăturii cu următorul modul prin cea de-a treia componentă, Client-Side Angular Services.

Cel de-al doilea modul, este structurat în patru componente care sunt responsabile pentru diferite activități de execuție. Controller-ul, este componenta care interacționează în mod direct cu primul modul și conține end-point-urile care sunt accesate de Client-Side Angular Services. În cea de-a doua componentă, Services, se realizează logica de afaceri bazată pe datele primite de la utilizator. JPA Repository este componenta care interacționează în mod direct cu baza de date, oferind funcții de manipulare a datelor utilizând reprezentările interne ale tabelor. Acestea sunt create cu ajutorul mapării entităților din componenta Entity.

Ultimul modul, DataBase Server este responsabil pentru stocarea, structurarea și organizarea datelor, astfel încât accesarea lor să fie posibilă într-un mod cât mai eficient.

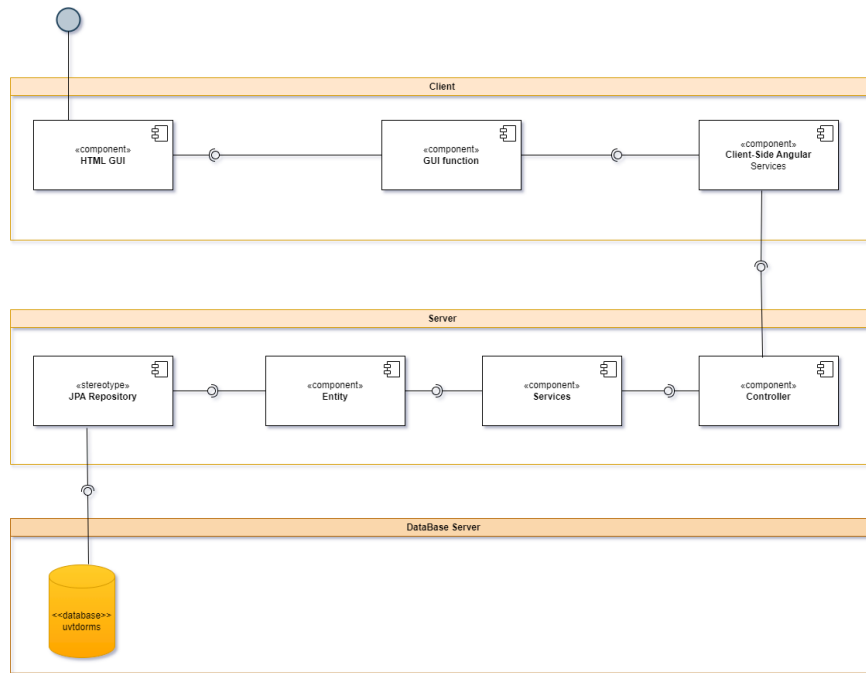


Figura 2.6: Diagrama unităților de execuție

2.3 Structura bazei de date

Având în vedere varietatea obiectelor abstracte pe care le folosește sistemul și necesitatea de reutilizare a acestora, aplicația își are datele structurate într-o bază de date. Toate elementele din această structură încapsulează informații esențiale despre principalele obiecte cu care operează aplicația.

Una dintre cele mai importante abstractizări este cea a userului, tabelul *users*, care este o structură ce încapsulează datele unui utilizator obișnuit, cum ar fi numele, adresa de e-mail, numărul de telefon, parola, imaginea de profil, rolul, starea de activitate a contului, evenimentele și, desigur, un identificator unic.

Printre tipurile de utilizatori se enumără și studenții, care au anumite date pe care restul tipurilor de utilizatori nu le au, motiv pentru care a fost nevoie de crearea unei tabele, *students_details*, care are ca atribute detaliile necesare lucrării cu obiecte ce abstractizează entitatea student. Printre aceste detalii se află camera, tichetele, programările și cererile de înregistrare.

Un alt tip de utilizator cu detalii diferite este administratorul căminului, care, pe lângă datele unui utilizator obișnuit, mai are și datele căminului pentru care este responsabil, dar și evenimentele create de acesta. Pentru încapsularea acestor informații am creat tabelul *dorm_administrator_details*.

Anunțurile din sistem sunt încapsulate în tabelul *events*, care conține un identificator unic al anunțului, administratorul care a creat evenimentul, un titlu, un text ce reprezintă descrierea anunțului, data în care anunțul a fost creat, data desfășurării evenimentului, posibilitatea de a putea sau nu participa la eveniment și studenții care vor participa.

Căminele aflate în sistem au fiecare un identificator unic, o adresă, un nume, camerele, mașinile de spălat, uscătoarele, dar și tichetele asociate, toate fiind încapsulate în tabelul *dorms*. Camerele din cămine au într-un mod similar un identificator unic,

numărul camerei, căminul, studenții locatari și cererile de înregistrare a studenților pentru camera respectivă, detalii prezente în tabelul *rooms*.

Datele mașinilor de spălat și ale uscătoarelor din spălătoriile căminelor sunt în tabelele *wash_machines* și *dryers*. Ambele au un identificator unic, la fel și identificatorul unic al căminului în care se află, un număr de identificare în contextul căminului și un câmp care reprezintă starea mașinii, adică disponibilitatea acesteia și programările pentru fiecare.

Programările la spălătoria căminului sunt reprezentate prin structura tabelului *laundry_appointments*. Fiecare programare are un identificator unic, precum și identificatorul unic al utilizatorului care a creat-o, dar și identificatoarele unice al mașinii de spălat și al uscătorului la care s-a făcut programarea. Începutul, finalul rezervării și statusul programării sunt informații care sunt încapsulate tot în această structură.

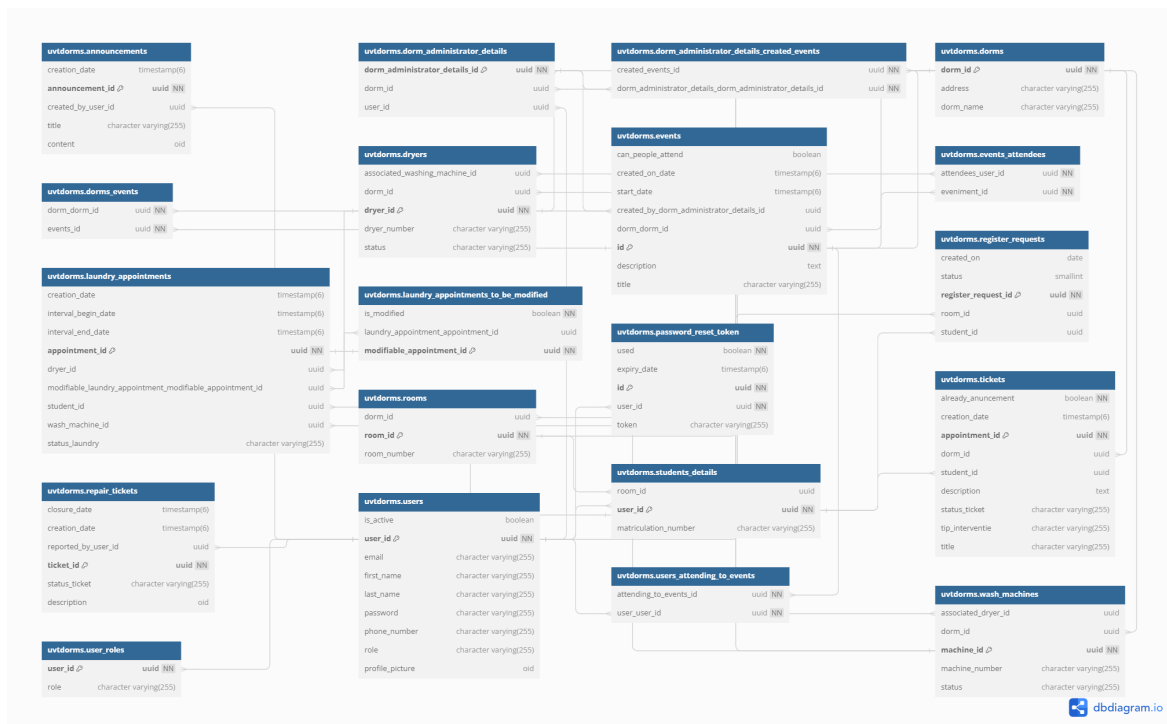


Figura 2.7: Diagrama unităților de cod pentru entități

Capitolul 3

Implementarea aplicației

Pentru realizarea propriu-zisă a aplicației urmăresc structura prezentată în capitolul anterior. Dezvoltarea se desfășoară în paralel atât pe partea de front-end a aplicației, cât și pe partea de back-end. Astfel, fiecare funcționalitate este implementată incremental, iar testarea lor este posibilă și din perspectiva utilizatorului, deja de la începutul dezvoltării.

Având în vedere că implementările unor mecanisme sunt similare, vor fi prezentate doar câte un exemplar din cele distincte. Deși pe parcursul realizării codului sursă se urmăresc principiile *Clean Code*[?] și se evită repetarea codului și se profită de avantajele programării orientate pe obiect, este totuși imposibilă modularizarea microserviciilor, astfel încât să nu existe părți similare.

3.1 Configurarea backend-ului

Printre lucrurile prioritare în dezvoltarea aplicației se află realizarea conexiunii dintre front-end și back-end. Aceasta presupune mai mulți factori, principalii fiind de natură de securitate. Framework-ul Spring-Boot permite variate configurații ale serverului pentru a asigura accesarea serviciilor doar de către cei autorizați, creând astfel măsurile de bază ale securității aplicației[?].

În secțiunile următoare vor fi prezentate configurarea accesului la serviciile serverului, atât prin condiționarea adreselor care solicită acces, cât și prin condiționarea accesului prin verificarea parametrilor din solicitare. Cea din urmă este executată doar în cazul anumitor puncte de acces, fiindcă serverul are și servicii *publice*, adică servicii care sunt accesibile nu numai utilizatorilor care au un cont.

3.1.1 Cross-Origin Resource Sharing

Unul dintre cele mai aplicate mecanisme de filtrarea apelurilor este *Cross-Origin Resource Sharing*[?] (CORS). Acesta vine cu funcționalități diferite în funcție de nivelul de aplicabilitate.

La nivelul protocolului, CORS adaugă un câmp nou în header-ul cererilor, care indică originea solicitării de acces. Astfel, pot fi restricționate originile de la care serverul să accepte cereri. Tot aici, pot fi specificate și tipul cererilor ce se pot face pentru back-end (GET, PUT, OPTIONS, etc.).

La nivel de API (Application Programming Interface) cererile restricționate vor primi totuși un răspuns, generat de browser. Un mesaj de eroare se generează în cazul

cererilor respinse, dar acesta nu este accesibil din script, doar apare în consolă.

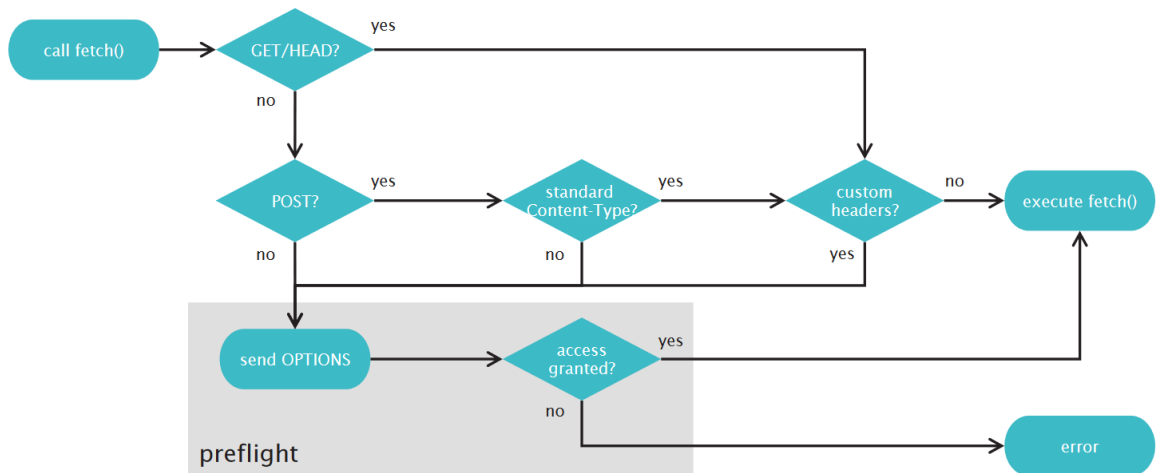


Figura 3.1: CORS flow[?]

Pentru a configura CORS pentru aplicația UVTDorms, se crează clasa ‘*CorsConfig*’ în folderul ‘*utils/*’. Clasa are un ‘*@Bean*’, adică o funcție ce returnează un obiect de tip ‘*CorsFilter*’. Aici este creată sursa ce permite configurarea CORS pe baza URL-ului. Configurarea acoperă aspecte cum ar fi:

1. configurarea originilor cu acces (pe parcursul dezvoltării, singurul URL cu acces este ‘*http://localhost:4200*’)
2. configurarea tipurilor de header permise
3. specificarea metodelor permise
4. setarea punctelor de acces pentru care se aplică configurarea CORS (pentru a include toate punctele, se folosește URL-ul ‘*/***’)

```
1 @Configuration
2 @EnableWebMvc
3 public class CorsConfig
4 {
5     @Bean
6     public CorsFilter corsFilter(){
7         UrlBasedCorsConfigurationSource source= new
8             UrlBasedCorsConfigurationSource();
9         CorsConfiguration config = new CorsConfiguration();
10        config.setAllowCredentials(true);
11        config.addAllowedOrigin("http://localhost:4200");
12        config.setAllowedHeaders(Arrays.asList(
13            HttpHeaders.AUTHORIZATION,
14            HttpHeaders.CONTENT_TYPE,
15            HttpHeaders.ACCEPT
16        ));
17        config.setAllowedMethods(Arrays.asList(
18            HttpMethod.GET.name(),
```

```

18         HttpMethod.POST.name(),
19         HttpMethod.PUT.name(),
20         HttpMethod.DELETE.name()
21     ));
22     config.setMaxAge(3600L);
23     source.registerCorsConfiguration("/**", config);
24
25     return new CorsFilter(source);
26 }
27 }

```

Listing 3.1: Clasa prin care se realizează configurarea CORS

3.1.2 JSON Web Token Authentication

Una dintre cele mai sigure metode de autentificare și de încapsulare a anumitor informații relevante este JSON Web Token[?]. Acesta este un token care, de fapt reprezintă un obiect JSON criptat. În mod normal, aceste obiecte conțin informații necesare la autentificarea utilizatorilor. Deoarece este criptat, token-ul poate fi salvat local, în cache-ul browser-ului și poate fi citit la următoarea accesare a aplicației web.

În cadrul aplicației UVTDorms, obiectul JSON criptat conține o cheie de autentificare temporară, rolul și adresa de email ale utilizatorului. Token-ul obținut prin criptarea obiectului este inclus în header-ul fiecărei solicitări ce se face din front- spre back-end. Acesta este generat în momentul în care utilizatorul se autentifică și selectează opțiunea de *‘Tine-mă minte’*.

```

1 public String createToken(TokenDto dto)
2 {
3     Date now = new Date();
4     Date validity = new Date(now.getTime() + 3_600_000);
5
6     return JWT.create()
7         .withIssuer(dto.getEmail())
8         .withIssuedAt(now)
9         .withExpiresAt(validity)
10        .withClaim("role", dto.getRole().toString())
11        .sign(Algorithm.HMAC256(secretKey));
12 }

```

Listing 3.2: Funcția de generare a token-ului

3.1.3 Security configuration

În cadrul aplicației UVTDorms, un alt strat de securitate este adăugat prin configurarea Spring Security[?]. Acesta este un framework ce oferă funcționalități de securitate pentru aplicațiile Java. Permite configurarea securității la nivel de URL, de metode HTTP, de acces la resurse și de autentificare. Face posibilă crearea de end-point-uri ‘publice’ și ‘private’, în funcție de necesități.

Pentru a configura Spring Security, se crează o clasă *‘SecurityConfig’* în folderul *‘utils/’*. Aceasta are o metodă care returnează un *‘SecurityFilterChain’*, ce conține configurarea end-point-urilor securizate. Configurarea include:

1. Verificarea de JWT înaintea accesării end-point-urilor

2. Crearea unei sesiuni STATELESS, adică fără stocarea informațiilor de autentificare
3. Crearea de excepții pentru cazurile în care token-ul nu este valid sau nu este prezent

```

1 @RequiredArgsConstructor
2 @Configuration
3 @EnableWebSecurity
4 public class SecurityConfig {
5
6     private final UserAuthProvider userAuthProvider;
7
8     @Bean
9     public SecurityFilterChain securityFilterChain(HttpSecurity http)
10        throws Exception {
11        http.csrf(AbstractHttpConfigurer::disable)
12            .addFilterBefore(new JwtAuthFilter(userAuthProvider),
13                BasicAuthenticationFilter.class)
14            .sessionManagement(customizer -> customizer.
15                sessionCreationPolicy(SessionCreationPolicy.STATELESS
16                ))
17            .authorizeHttpRequests((requests) -> requests.
18                requestMatchers(HttpMethod.POST, "/api/auth/login")
19                    .permitAll()
20                    .requestMatchers(HttpMethod.GET, "/api/dorms/
21                        dorms-names").permitAll()
22                    .requestMatchers(HttpMethod.POST, "/api/auth/
23                        register-student").permitAll()
24                    .requestMatchers(HttpMethod.GET, "/api/rooms/get-
25                        rooms-numbers-from-dorm/**").permitAll()
26                    .anyRequest().authenticated());
27        return http.build();
28    }
29 }

```

Listing 3.3: Clasa prin care se realizează configurarea Spring Security

Configurarea creată permite accesul fără token pentru end-point-urile de autentificare, înregistrare și pentru accesarea listei de cămine. Pentru toate celelalte end-point-uri, accesul este permis doar utilizatorilor autentificați.

3.1.4 Exception handler

Pentru standardizarea modului de gestionare a excepțiilor, am creat un set de clase care permit tratarea excepțiilor într-un mod unitar. În primul rând, am creat o clasă ‘*AppException*’, care extinde clasă ‘*RuntimeException*’ și este o clasă de bază pentru toate excepțiile aplicației. Aceasta conține un mesaj și un status *HTTP*.

```

1 public class AppException extends RuntimeException {
2     private final HttpStatus httpStatus;
3
4     public AppException(String message, HttpStatus httpStatus) {
5         super(message);
6         this.httpStatus = httpStatus;
7     }
8 }

```

```

8
9     public HttpStatus getHttpStatus() {
10         return httpStatus;
11     }
12 }

```

Listing 3.4: Clasa de bază pentru excepții

Acest mesaj de eroare este pus într-un obiect de tip *‘ErrorDto’*, care este un *record*[?] ce încapsulează mesajul de tip *String*. Acesta este folosit pentru a transmite mesajul de eroare în body-ul unui răspuns de tip *‘ResponseEntity’*.

```

1 public record ErrorDto(String message) {}

```

Listing 3.5: Record pentru mesaje de eroare

Pentru a gestiona automat orice apariție de excepție de tipul *‘AppException’*, am creat un *‘ControllerAdvice’*, care detectează excepțiile de acest tip, crează un obiect de tip *‘ErrorDto’* și îl returnează într-un răspuns de tip *‘ResponseEntity’*.

```

1 @ControllerAdvice
2 public class RestExceptionHandler {
3     @SuppressWarnings("null")
4     @ExceptionHandler(value = { AppException.class })
5     @ResponseBody
6     public ResponseEntity<ErrorDto> handleException(AppException ex)
7     {
8         return ResponseEntity.status(ex.getHttpStatus())
9             .body(new ErrorDto(ex.getMessage()));
10    }
11 }

```

Listing 3.6: Clasă care gestionează excepțiile apărute

3.2 Configurarea bazei de date

Pentru stocarea datelor aplicației UVTDorms, am ales să folosesc un sistem de gestionare de baze de date relaționale, și anume PostgreSQL[?]. Arhitectura bazei de date poate fi observată în capitolul??.

3.2.1 JPA Repository

Pentru a accesa baza de date, am folosit JPA Repository[?]. Acesta oferă un strat de abstractizare dintre o bază de date și aplicația Spring-Boot. Prin intermediul JPA Repository, se pot crea ușor metode de manipulare a datelor, precum adăugarea, ștergerea sau modificarea acestora. Acest strat oferă și posibilitatea de a crea metode specifice pentru operații complexe, dar permite și crearea de metode de căutare după anumite criterii, cum ar fi numele sau adresa de email a unui utilizator, fără scrierea de query-uri SQL.

Pentru a crea un repository, se crează o interfață care extinde *‘JpaRepository’* și care are ca parametri tipul entității și tipul cheii primare a acesteia. În această interfață se pot adăuga metode specifice pentru operații complexe. În exemplul următor poate fi observat implementarea unor metode specifice pentru entitatea *‘Room’*.

```

1 @Repository
2 public interface IRoomRepository extends JpaRepository<Room, Long> {
3     public Optional<Room> getRoomByRoomNumber(String roomNumber);
4     public List<Room> findByDormDormName(String dormName);
5     public Optional<Room> findByDormAndRoomNumber(Dorm dorm, String
        roomNumber);
6     public Optional<Room> findByDormDormNameAndRoomNumber(String dormName
        , String roomNumber);
7 }

```

Listing 3.7: Interfața JPA Repository pentru entitatea Room

3.2.2 Maparea claselor Spring-Boot la tabelele PostgreSQL

Prin intermediul JPA Repository, se poate realiza maparea entităților (claselor Java) la tabelele din baza de date. Această mapare se realizează prin numele membrilor claselor și cu ajutorul adnotărilor. Adnotările oferă informații suplimentare despre cum să fie interpretate clasele și membrii acestora. Tot aici se pot specifica și relațiile dintre entități.

În exemplul de mai jos se poate observa cum se realizează maparea entității *'User'* la tabela *'users'*. Entitatea are un identificator unic de tip UUID, numele, prenumele, adresa de email, numărul de telefon, parola și alte detalii specifice unui utilizator. De asemenea, entitatea are și relații cu alte entități, precum *'StudentDetails'* sau *'DormAdministratorDetails'*.

```

1 @Getter
2 @Setter
3 @NoArgsConstructor
4 @AllArgsConstructor
5 @Entity
6 @Builder
7 @Table(name = "users")
8 public class User {
9     @Id
10    @GeneratedValue(generator = "UUID")
11    private UUID userId;
12    private String firstName;
13    private String lastName;
14    @Column(unique = true)
15    private String email;
16    @Column(unique = true)
17    private String phoneNumber;
18    private String password;
19
20    @OneToOne(mappedBy = "user", cascade = CascadeType.ALL)
21    private StudentDetails studentDetails;
22
23    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
24    private List<RepairTicket> repairTickets;
25
26    @Enumerated(EnumType.STRING)
27    private Role role;
28
29    @OneToOne(mappedBy = "administrator", cascade = CascadeType.ALL)
30    private DormAdministratorDetails dormAdministratorDetails;

```

```

31
32     @ManyToMany
33     private List<Eveniment> attendingToEvents;
34
35     private Boolean isActive;
36
37     @Lob
38     private byte[] profilePicture;

```

Listing 3.8: Clasa User

3.2.3 Data Transfer Object

Datele aplicației UVTDorms sunt stocate în entitățile mapate la tabelele din baza de date. Acestea conțin toate informațiile asociate unei entități, inclusiv relațiile cu alte entități. De fiecare dată când front-end-ul solicită informații serverului, acestea sunt trimise sub formă de obiecte. Pentru a evita trimiterea unui obiect întreg și pentru a restructura informațiile într-un mod mai ușor de folosit, am creat obiecte de transfer de date (DTO)[?].

Aceste obiecte sunt clase simple, care conțin doar informațiile necesare pentru o anumită operație. De exemplu, pentru a afișa informațiile unui utilizator pe pagina de profil, se folosește clasa *UserDetailsDto*, care conține doar numele, prenumele, adresa de email și numărul de telefon al utilizatorului.

```

1 public class UserDetailsDto {
2     private String firstName;
3     private String lastName;
4     private String email;
5     private String phoneNumber;
6 }

```

Listing 3.9: Clasa UserDetailsDto

3.3 Servicii back-end

Serviciile back-end sunt clase care reprezintă partea logică a aplicației. Acestea sunt responsabile pentru manipularea datelor, pentru efectuarea operațiilor complexe și pentru pregătirea datelor pentru a fi trimise către front-end. Fiecare set de funcționalități ale aplicației are propriul serviciu, care este responsabil pentru operațiile specifice.

Serviciile pot să comunice atât cu alte servicii cât și cu baza de date. În cazul în care un serviciu are nevoie de date, acesta apelează metodele din repository pentru a obține informațiile necesare. În cazul în care serviciul are nevoie de date de tipul unei entități neasociate serviciului, se apelează metodele din repository-urile altor entități pentru a obține informațiile necesare. Dacă aceste date trebuie să fie prelucrate, se pot apela serviciile care se ocupă de aceste operații.

De exemplu, clasa *LaundryAppointmentService* oferă un serviciu pentru crearea programărilor la spălătorie, prin intermediul metodei *createLaundryAppointment(...)*. Acest serviciu se ocupă doar de operațiile legate de programări. În cadrul acestei metode se întâmplă următoarele:

1. se verifică dacă studentul are deja o programare pentru aceeași săptămână,
2. se verifică dacă mașina de spălat și uscătorul sunt disponibile,
3. se procesează intervalul selectat de student, astfel încât să fie în formatul necesar creării programării,
4. se creează programarea,
5. programarea se salvează în baza de date.

```

1  @Service
2  @RequiredArgsConstructor
3  public class LaundryAppointmentService {
4      // ... class members and private helper functions ...
5
6      public void createLaundryAppointment(CreateLaundryAppointmentDto
7          createLaundryAppointmentDto, String studentEmail) throws
8          AppException {
9          User user = userRepository.getByEmail(studentEmail)
10             .orElseThrow(() -> new AppException("User not found",
11                 HttpStatus.NOT_FOUND));
12
13             StudentDetails student = studentDetailsRepository.findByUser(user)
14                 .orElseThrow(() -> new AppException("The user is not a student",
15                     HttpStatus.BAD_REQUEST));
16
17             if (studentAlreadyHasAppointmentForThisWeek(student,
18                 createLaundryAppointmentDto.selectedDate()
19                     .atTime(createLaundryAppointmentDto.selectedInterval(), 0))) {
20                 throw new AppException("The student already has an appointment for this week",
21                     HttpStatus.BAD_REQUEST);
22             }
23
24             WashingMachine washingMachine = washingMachineRepository
25                 .findById(createLaundryAppointmentDto.selectedMachineId())
26                 .orElseThrow(() -> new AppException("Washing machine not found",
27                     HttpStatus.NOT_FOUND));
28
29             Dryer dryer = dryerRepository.findById(
30                 createLaundryAppointmentDto.selectedDryerId())
31                 .orElseThrow(() -> new AppException("Dryer not found",
32                     HttpStatus.NOT_FOUND));
33
34             LocalDateTime intervalBeginDate = createLaundryAppointmentDto.selectedDate()
35                 .atTime(createLaundryAppointmentDto.selectedInterval(), 0);
36             LaundryAppointment laundryAppointment = new LaundryAppointment(
37                 intervalBeginDate, student,
38                 washingMachine, dryer);
39             laundryAppointmentRepository.save(laundryAppointment);
40         }
41     }

```

```

34 // ... other methods ...
35 }

```

Listing 3.10: Clasa LaundryAppointmentService

3.4 Accesarea serviciilor back-end

Accesarea serviciilor back-end nu se poate realiza în mod direct, ci prin intermediul unor end-point-uri. Acestea sunt metode ce sa află în stratul controller al aplicației și sunt responsabile pentru a primi cererile de la front-end și pentru a le redirecționa către serviciile corespunzătoare.

3.4.1 Rest Controller și maparea resurselor

Pentru a crea un end-point accesibil din front-end, se crează o clasă care are adnotările ‘*@RestController*’ și ‘*@RequestMapping*’[?]. Acestea indică faptul că clasa este un controller și că metodele acesteia sunt accesibile prin intermediul unor URL-uri specifice.

Controllerele au ca attribute serviciile la care transmit cererile primite. Acestea apelează metodele serviciilor și returnează rezultatele către front-end. În cazul în care o cerere nu este validă, controllerul poate să arunce o excepție, care va fi prinsă de un ‘*ControllerAdvice*’?? și va fi transformată într-un răspuns de eroare.

De exemplu, controllerul responsabil pentru accesarea serviciilor aferente uscătoarelor din spălătoria unui cămin este ‘*DryerController*’.

```

1 @RestController
2 @RequestMapping("/api/dryers")
3 @RequiredArgsConstructor
4 public class DryerController {
5     private final DryerService dryerService;
6
7     // ... other methods ...
8 }

```

Listing 3.11: Clasa DryerController

3.4.2 Configurarea end-point-urilor

Pentru configurarea end-point-urilor se folosește adnotarea ‘*@GetMapping*’, care este specifică metodei HTTP GET, și adnotarea ‘*@PostMapping*’, care este specifică metodei HTTP POST[?]. Acestea sunt folosite pentru a specifica URL-ul la care este accesibilă metoda.

Metodele pot primi parametri de la front-end, care sunt extrase atât din header-ul cererii, cât și din body-ul acesteia. În cazul în care parametrii sunt trimiși în body-ul cererii, aceștia sunt obținuți prin anotarea ‘*@RequestBody*’. În cazul în care parametrii sunt trimiși în URL-ul cererii, aceștia sunt obținuți prin anotarea ‘*@PathVariable*’. Iar în cazul în care informațiile se află în header-ul cererii, cum ar fi token-ul de autentificare, acesta se extrage separat și controllerul o primește la parametri ca și obiect de tip ‘*Authentication*’.

De exemplu, o metodă a controllerului ‘*DryerController*’ este ‘*getDryersFromDorm*’, care primește numele unui cămin și returnează lista de uscătoare din acel cămin.

```
1 @GetMapping("/get-dryer-from-dorm/{dormId}")
2 public ResponseEntity<List<DryerDto>> getDyerFromDorm(@PathVariable("
   dormId") String dormId) {
3     return ResponseEntity.ok(this.dryerService.getDyerFromDorm(dormId));
4 }
```

Listing 3.12: Metoda *getDryersFromDorm*

Un alt exemplu de metodă, care primește ca parametru un obiect de tip ‘*Authentication*’, este metoda ‘*getUserDetails (...)*’ din controllerul ‘*UserController*’, care returnează detaliile unui utilizator. Acesta primește token-ul de autentificare care este extras din header-ul cererii, pentru a-l identifica pe utilizator.

```
1 @GetMapping("/get-user-details")
2 public ResponseEntity<UserDetailsDto> getUserDetails(Authentication
   authentication) {
3     TokenDto userToken = (TokenDto) authentication.getPrincipal();
4     UserDetailsDto userDetailsDto = userService.getUserDetails(userToken.
   getEmail());
5     return ResponseEntity.ok(userDetailsDto);
6 }
```

Listing 3.13: Metoda *getUserDetails(...)*

3.5 Configurarea serviciului de Email

Serviciile email reprezintă o parte importantă a aplicației UVTDorms, acestea fiind principalul canal de comunicare. Am ales să transmit toate informațiile importante prin email, fiindcă astfel mesajele pot ajunge și la utilizatorii care nu urmăresc în mod activ fluxul de activitate al aplicației.

Pentru a utiliza acest serviciu, m-am folosit de o clasă ce aparține pachetului ‘*java-mail*’ din framework-ul ‘*Spring Boot*’. Acesta oferă o metodă simplă și ușoară pentru compunerea și trimiterea email-urilor, prin setarea unor câmpuri esențiale, cum ar fi subiectul și receptorul[?].

Pentru a trimite email-uri care au o structură HTML, ceea ce permite stilizarea și formatarea email-urilor, am folosit ‘*SpringTemplateEngine*’, care este un pachet ce aparține de ‘*Thymeleaf*’. Cu ajutorul acestuia se pot deschide fișiere HTML șablonizate (în care se specifică variabile ce urmează să fie înlocuite în momentul aplicării șablonului). Pentru a înlocui variabilele din șablon, se specifică cuvântul cheie unic prin care se identifică fiecare variabilă și se specifică înlocuitorul acesteia[?].

Astfel, aplicația UVTDorms trimite email-uri utilizatorilor cu toate informațiile relevante. Tot prin email se primesc confirmările de înregistrări ale conturilor (atât la studenți cât și la administratorii de cămin), cu parola nouă generată, dar și notificările, cum ar fi anularea unei programări, se trimit tot prin email.

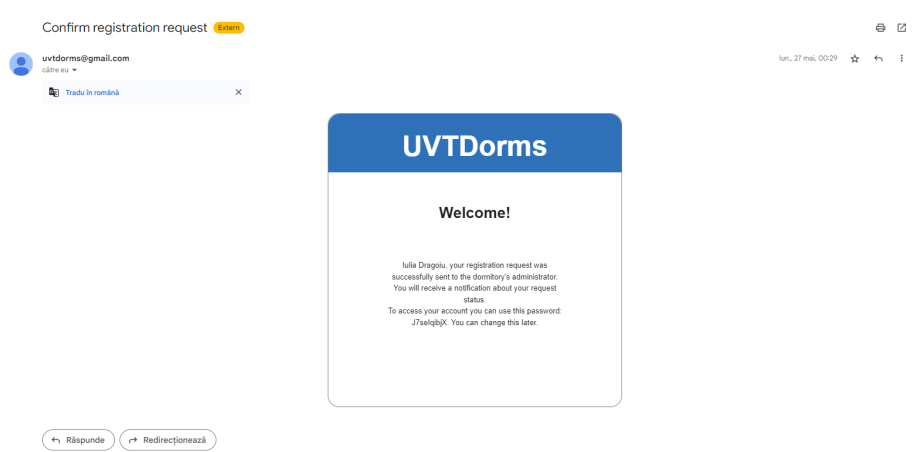


Figura 3.2: Exemplu email de confirmare

3.6 Componentele Angular

Interfața aplicației, partea de front-end, este construită cu Angular, care este un framework bazat pe TypeScript[?]. Acest framework permite alcătuirea paginilor și a componentelor într-un mod optim, oferind un mediu de dezvoltare ușor de menținut.

Framework-ul Angular permite crearea componentelor, fiecare având câte un fișier de TypeScript, HTML, CSS și testare. Astfel, componentele sunt foarte bine modularizate. În aplicația UVTDorms, fiecare pagină este o componentă separată, cu propriile funcționalități. Unele pagini au propriile lor subcomponente în cazul în care complexitatea funcționalităților și, respectiv sau, aspectul reutilizabilității solicită acest lucru.

Accesarea paginilor este condiționată de rolul utilizatorului. În cazul în care un utilizator neautentificat intră pe aplicația web, singura pagină accesibilă acestuia este pagina de autentificare. Orice încercare de a intra pe o altă pagină va eșua, fiindcă verificarea rolului se întâmplă în core-ul aplicației. În mod similar, unele subcomponente și elemente apar doar pentru anumiți utilizatori. De exemplu, în bara de navigare a aplicației, butoanele de navigare apar în funcție de rolul utilizatorului. Astfel, pentru un anumit tip de utilizator, apar doar butoanele care duc la paginile la care utilizatorul are acces.

În cazul paginilor comune, cum ar fi pagina utilizatorului, unde detaliilor contului diferă, din baza de date se vor cere doar datele potrivite rolului utilizatorului, fără să se mai apeleze și end-point-urile pentru datele altor roluri.

3.6.1 Biblioteci de componente IU

Angular permite utilizarea bibliotecilor de componente de interfețe de utilizator. Aceste biblioteci oferă componente costumizabile, atât pe partea de afișare cât și pe partea de funcționalități. În mod normal, utilizarea unor astfel de biblioteci este o soluție eficientă, fiindcă permit o dezvoltare mai rapidă și consistentă.

Pentru dezvoltarea aplicației UVTDorms, am folosit două astfel de biblioteci: *'Material UI'*[?] și *'PrimeNG'*[?]. Prima este o bibliotecă dezvoltată chiar de Angular. A doua este independentă, dar este mult mai mare și oferă componente mai complexe. Eu am folosit componentele din aceste biblioteci în special în form-uri, elementele lor

de input având un aspect foarte bine realizat. Bibliotecile s-au dovedit a fi foarte eficiente și la afișarea datelor, cum ar fi în cazul tabelelor.

3.7 Sistemul de rutare în front-end

Angular are propriul sistem de rutare, care permite navigarea dintre paginile aplicației web fără reîncărcarea paginii în browser. Configurarea sistemului de rutare presupune specificarea path-ului, cum ar fi `/home` și asocierea acesteia cu o componentă Angular, cum ar fi `HomePageComponent`. Astfel, când utilizatorul accesează un path al paginii, este încărcată componenta asociată. Navigarea internă presupune schimbarea path-ului și afișarea componentei asociate, fără reîncărcarea paginii.

Pentru a configura sistemul de rutare, Angular are o componentă dedicată, `AppRoutingModule`, care are o listă de path-uri asociate cu componentele. Aplicația UVT-Dorms are 17 pagini configurate în această listă.

3.8 Servicii front-end

În scopul modularizării și structurării potrivite a codului, Angular are un mecanism pentru a separa serviciile prin care trec cererile la back-end, de componentele aplicației. Astfel, toate apelurile au loc în servicii dedicate, fiecare parte a aplicației având propriul serviciu. De exemplu, camerele de cămin au un serviciu care conține toate funcțiile cu care pot fi apelate end-pointurile din serviciul de camere al back-endului. Prin urmare, fiecare serviciu de back-end are un serviciu asociat în front-end.

În mod normal, un serviciu în front-end este alcătuit din constante de string-uri care reprezintă path-urile la end-pointurile serviciul asociat din back-end și funcțiile prin care se realizează apelurile la aceste end-pointuri. Funcțiile acestea sunt folosite de restul componentelor.

O parte importantă a serviciilor este configurarea mesajului ce urmează să fie trimis. După cum am descris în capitolele anterioare, UVT-Dorms are un sistem de JWTToken pentru autentificare. Acest token trebuie să fie inclus în header-ul apelurilor spre back-end, iar acest lucru se întâmplă în servicii, unde se apelează un serviciu special, `AuthService`, care are responsabilitatea de a gestiona token-ul salvat. Prin urmare, serviciul `AuthService` este prezent aproape în toate serviciile front-end al aplicației și are rolul de a furniza header-ul necesar pentru comunicarea cu back-endul.

Un alt serviciu puțin mai special decât restul serviciilor este `UserService`, care, pe lângă structura generală a serviciilor front-end (constante de path-uri și funcții de apelare), are responsabilitatea de a stoca rolul utilizatorului. Acest serviciu este folosit în toate componentele care au un comportament dependent de tipul de utilizator.

3.9 Internaționalizare

Având în vedere că Universitatea de Vest din Timișoara nu are doar studenți vorbitori de limba română, aplicația UVT-Dorms pune la dispoziția utilizatorilor interfața atât în limba română cât și în limba engleză.

Datorită Angular, implementarea unei interfețe multilingv se poate realiza eficient, utilizând un pachet special pentru traducerea aplicațiilor web, `ngx-translate`[?]. Cu

ajutorul acestui pachet, traducerea aplicației se realizează prin crearea a câte un fișier JSON pentru fiecare limbă. În aceste fișiere, se specifică cuvinte cheie care se asociază cu un string. În HTML-ul aplicației se specifică cuvântul cheie din JSON și textul afișat va fi în limba selectată de utilizator.

Pentru a schimba limba interfeței, utilizatorul are în bara de navigare un buton care este steagul țării în a cărei limbă este afișarea curentă. Apăsând pe buton, utilizatorul poate schimba între limbile română și engleză.

Capitolul 4

Ghid utilizator

În acest capitol este prezentat ghidul de utilizare al aplicației din perspectiva fiecărui tip de utilizator.

4.1 Autentificare

Pagina de pornire a aplicației UVTDorms este pagina de autentificare. Aici utilizatorul are posibilitatea de a se autentifica, de a se înregistra și de a-și recupera parola uitată.

Inițial, pe pagina de autentificare se află un formular de autentificare, unde utilizatorul poate să-și introducă adresa de email și parola. Apăsând pe butonul de autentificare, utilizatorul este redirecționat pe pagina contului său. În cazul în care adresa de email sau parola nu au fost corecte, un mesaj de eroare este afișat.

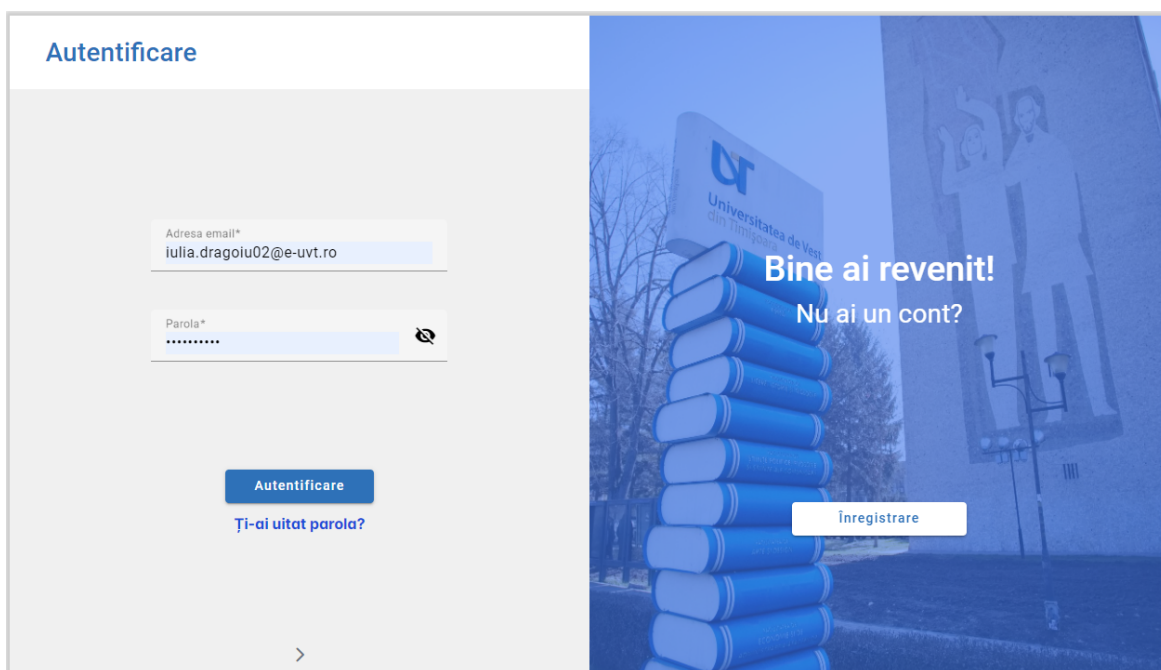


Figura 4.1: Formular de autentificare

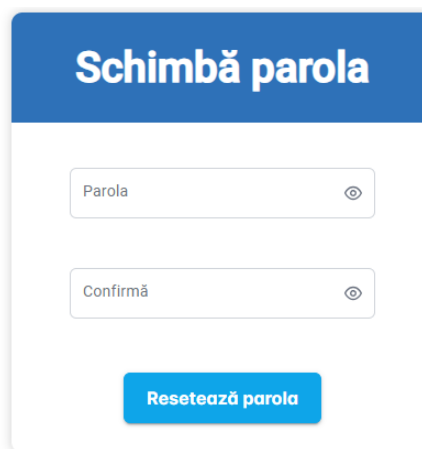
În cazul în care utilizatorul și-a uitat parola, apăsând pe butonul de *‘Ți-ai uitat*

parola?’, este redirectionat pe o altă pagină cu un formular unde poate să-și introducă adresa de mail. Apăsând pe butonul ‘*Trimite*’, i se trimite un mail cu un link către o pagină cu un formular unde poate să-și seteze o parolă nouă.



Formular de resetare parolă email. Titlu: **Îți venim în ajutor**. Text: Introduce adresa de email și îți vom trimite un link pentru a-ți reseta parola. Câmp de text: iulia.dragoiu02@e-uvt.ro. Buton: Trimite.

Figura 4.2: Formular resetare parolă email

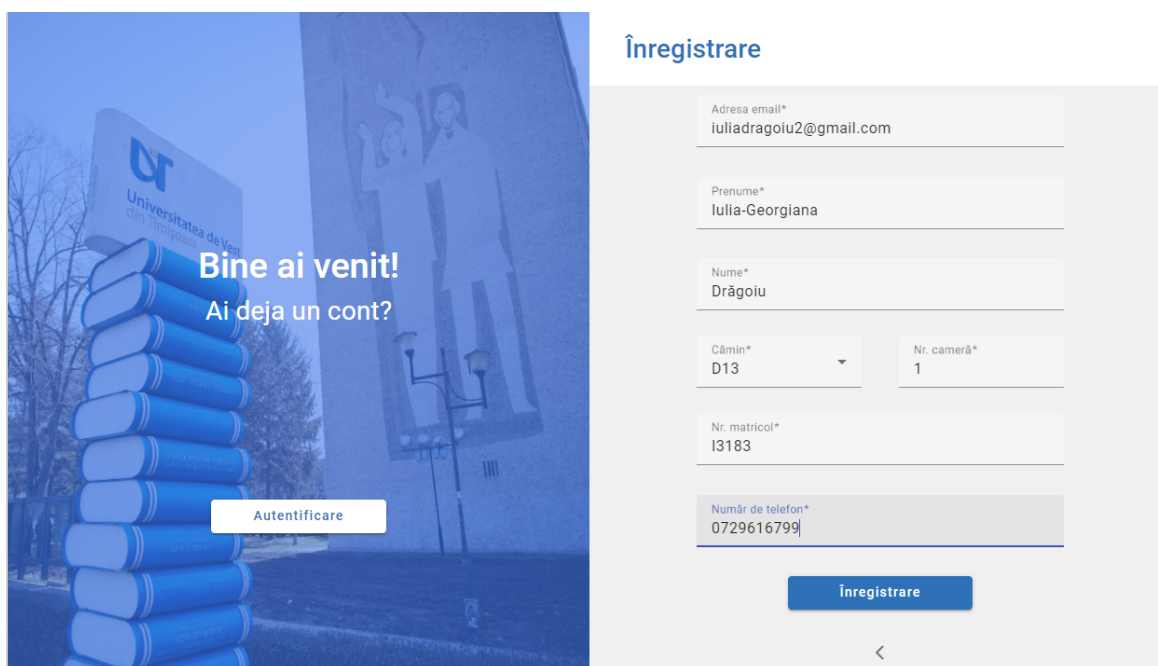


Formular de resetare parolă. Titlu: **Schimbă parola**. Câmpuri: Parola, Confirmă. Buton: Resetează parola.

Figura 4.3: Formular resetare parolă

4.1.1 Înregistrare

Studentii noi care încă nu au cont, pot să deschidă fereastra de înregistrare apăsând pe butonul ‘*Înregistrare*’. Aici trebuie să introducă datele personale, cum ar fi numele, adresa de email, numărul de telefon. Deodată cu înregistrarea contului se face și o cerere de înregistrare la cămin. Din acest motiv, în formularul de înregistrare studenții trebuie să selecteze și căminul și camera lor.



Formular de înregistrare. Titlu: **Înregistrare**. Câmpuri: Adresa email* (iuliadragoiu2@gmail.com), Prenume* (Iulia-Georgiana), Nume* (Dragoiu), Cămin* (D13), Nr. cameră* (1), Nr. matricol* (I3183), Număr de telefon* (0729616799). Buton: Înregistrare. Text: Bine ai venit! Ai deja un cont? Autentificare.

Figura 4.4: Formular de înregistrare

După trimiterea formularului, studenții vor primi un email cu parola generată (pe care pot să o schimbe ulterior). Acești studenți vor fi inactivi până când cererea lor nu va fi acceptată de către administratorul căminului. Astfel, singura fereastră la care vor avea acces după ce s-au autentificat, este fereastra cererilor de înregistrare la cămine.

4.1.2 Setări cont

Pe pagina de profil a utilizatorilor, aceștia au mai multe tipuri de setări posibile. Apăsând pe poza de profil, utilizatorii o pot schimba.

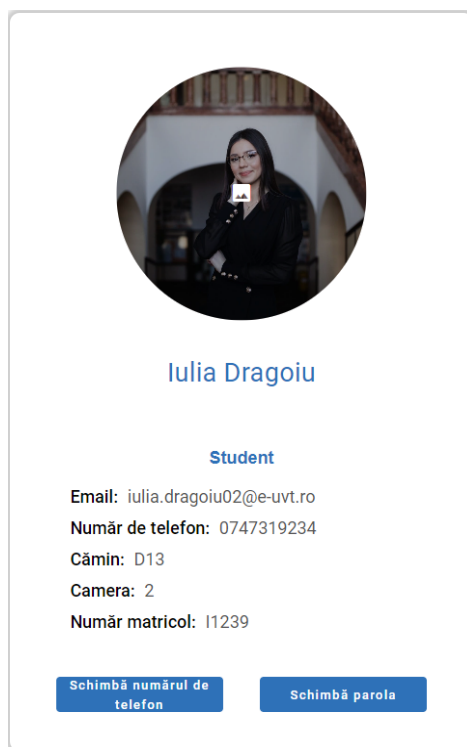


Figura 4.5: Detalii utilizator

În partea de jos a detaliilor utilizatorilor sunt două butoane:

- ‘*Schimbă numărul de telefon*’, cu care utilizatorul poate deschide un formular pentru schimbarea numărului de telefon
- ‘*Schimbă parola*’, cu care utilizatorul poate deschide formularul de schimbare a parolei

Schimbă nr. de telefon

Număr de telefon*

0729554047

Salvează

Figura 4.6: Formular de schimbare a numărului de telefon

Schimbă parola

Parola nouă*

Confirmă parola*

Salvează

Figura 4.7: Formular de schimbare a parolei

4.2 Student

În subsecțiunea următoare este prezentat ghidul de utilizare pentru student.

4.2.1 Cerere de înregistrare la cămin

Pe pagina de profil, studenții au o subfereastră ‘*Cereri de înregistrare*’. Aici au posibilitatea de a face cereri noi de înregistrare, apăsând pe butonul ‘+’ și selectând căminul și camera. După ce cererea a fost trimisă, aceasta apare în lista de cereri de înregistrare cu detaliile cererii, cum ar fi adresa de email a administratorului care trebuie să accepte cererea și statusul cererii.

Data creării cererii	Cămin	Camera	Contact	Status
2023-5-1	D13	1	tom.hanks@e-uvr.ro	ACCEPTED
2022-6-14	D13	3	tom.hanks@e-uvr.ro	DECLINED
2023-5-1	D13	2	tom.hanks@e-uvr.ro	ACCEPTED

Showing 1 to 3 of 3 requests

Figura 4.8: Tabelul cu cererile de înscriere a unui student

4.2.2 Crearea de programare la spălătorie

Studenții pot face câte o programare pe săptămână la spălătoria căminului. Pentru a face o programare nouă, pot intra pe pagina de programări (cu butonul ‘*Programări*’ din bara de navigare), unde au un formular de completat. Aici trebuie să aleagă ziua săptămânii (începând cu ziua curentă, inclusiv), mașina de spălat și intervalul orar.

Dacă formularul a fost completat, studentul trebuie să apese butonul ‘*Programează-te*’ și să confirme programarea.

Figura 4.9: Formularul de programare la spălătorie

4.2.3 Gestionarea programărilor existente

Pe pagina de profil, în subfereastra *‘Programările mele la spălătorie’* studentul are posibilitatea de a vedea toate programările sale: cele noi, cele anulate și cele deja trecute. Pentru cele noi studentul are posibilitatea de a le anula, apăsând pe butonul ‘X’.

T

The screenshot shows the 'Cămine' student portal. The user is 'Iulia Dragoiu', a student. The page has tabs for 'Programări', 'Tichete', and 'Evenimente'. The 'Programări' tab is active, showing a table of laundry appointments.

Mașina de spălat	Uscător de rufe	Data	Status	Anulează
Machine1	Dryer1	2024/6/16 10:00	SCHEDULED	X
Machine3		2024/5/25 10:00	COMPLETED	
Machine2	Dryer2	2024/6/1 10:00	COMPLETED	

Showing 1 to 3 of 3 requests

Figura 4.10: Programări la spălătorie

4.2.4 Crearea de tichete de raportare a problemelor

Pe pagina de tichete studentul poate completa un formular pentru anunțarea problemelor din cămin. Aici completând câmpurile de *‘Titlu’*, *‘Descriere’*, selectând tipul de intervenție necesar și specificând dacă problema a fost anunțată anterior, studentul poate trimite tichetul către administratorul căminului.

The screenshot shows a web form titled "Tickets". It contains the following fields and elements:

- Titlu**: A text input field with the placeholder text "Ușa nu se închide".
- Descriere**: A rich text editor with bold, italic, and underline icons. The text "Ușa nu se închide are nevoie de niște suruburi noi" is entered, with some words highlighted in red.
- Tip intervenție**: A dropdown menu with "TAMPLAR" selected.
- A checkbox labeled "Problema a fost anunțată în trecut?".
- A blue "Trimite" button at the bottom.

Figura 4.11: Formular de creare tichet

4.2.5 Lista de evenimente

Pe pagina de evenimente studenții și administratorii de cămin pot vizualiza lista de evenimente din cămin. Aici pot vedea titlul evenimentului, autorul, descrierea, data când va avea loc acesta și câți participanți vor fi. La fiecare eveniment apare și un buton ‘*Participă*’, cu care utilizatorii pot comunica intenția de a participa la eveniment. După apăsarea butonului, acesta se transformă în butonul ‘*Anulează participarea*’.

The screenshot displays a list of two events, each in a card format:

- Plant Swap Party: Grow Your Collection!**
 - Author:** Tom Hanks
 - Description:** Calling all plant lovers! Join us for a fun and sustainable Plant Swap Party. Bring a healthy, unwanted plant from your collection and swap it for something new! It's a great way to expand your plant family, declutter your space, and meet other plant enthusiasts. We'll also have resources and tips on plant care available.
 - What to Bring:**
 - A healthy, unwanted plant (pots included)
 - Your enthusiasm for all things green!
 - Action Button:** Anulează participarea (orange)
 - Participants:** 1 Participanți
 - Start Date:** Începe la 2024/07/05 21:08
- Volunteer Bake Sale: Supporting the Animal Shelter**
 - Author:** Tom Hanks
 - Description:** Calling all bakers and dessert enthusiasts! We're hosting a Volunteer Bake Sale to raise funds for the local animal shelter. Put your baking skills to the test and donate your delicious creations (or simply come by to indulge!). All proceeds will go towards providing care and comfort for homeless animals.
 - How to Participate:**
 - Sign up to donate baked goods by contacting
 - Donations can be dropped off at the event location on the day of the sale.
 - Action Button:** Participă (blue)
 - Participants:** 0 Participanți
 - Start Date:** Începe la 2024/08/14 21:08

Figura 4.12: Pagina cu evenimentele dintr-un cămin

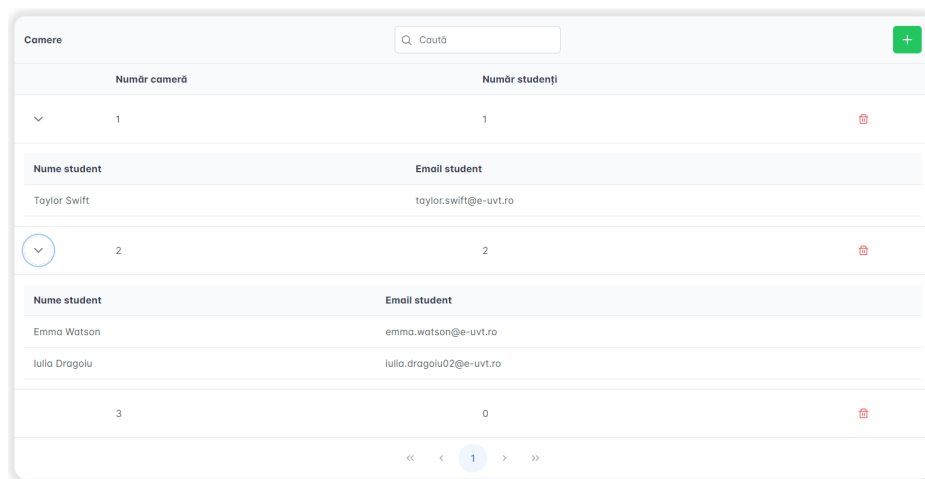
4.3 Administrator cămin

În subsecțiunea următoare este prezentat ghidul de utilizare pentru administratorul de cămin.

4.3.1 Administrarea camerelor

Administratorul căminului are responsabilitatea gestionării camerelor din cămin. Pentru a face acest lucru într-un mod eficient, UVTDorms pune la dispoziția administratorilor de cămine o pagină dedicată. Aici sunt listate toate camerele din cămin într-un tabel, unde apar și detalii, cum ar fi numărul camerei și numărul de studenți din cameră.

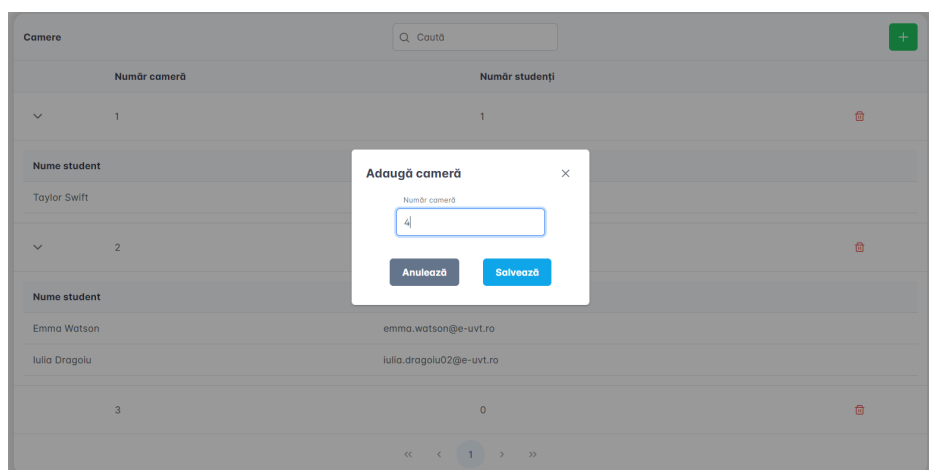
În partea stânga a tabelului, pentru fiecare cameră este un buton care deschide lista de studenți din camera respectivă, cu nume și adresă de email. În partea dreaptă a camerelor se află butonul de ștergere. Camerele pot fi șterse doar în cazul în care nu au niciun student.



Camere		
	Număr cameră	Număr studenți
▼	1	1
Nume student		
Taylor Swift		
Email student		
taylor.swift@e-uvt.ro		
▼	2	2
Nume student		
Emma Watson		
Email student		
emma.watson@e-uvt.ro		
Iulia Dragolu		
Email student		
iulia.dragolu02@e-uvt.ro		
	3	0

Figura 4.13: Tabelul camerele dintr-un cămin

Pentru a adăuga o cameră, administratorul trebuie să apese butonul verde '+', care deschide un formular unde trebuie specificat numărul camerei care urmează să fie adăugată.



Adaugă cameră ✕

Număr cameră

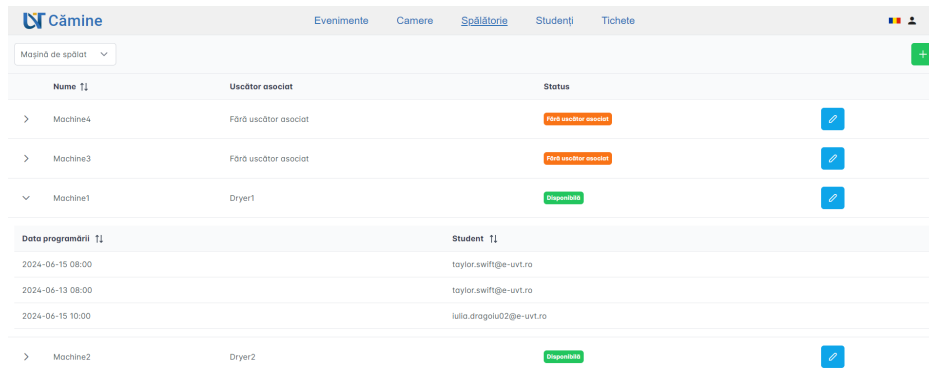
4

Anulează Salvează

Figura 4.14: Formular de adăugare de cameră

4.3.2 Administrarea spălătoriei

Administrarea spălătoriei este una dintre responsabilitățile principale ale administratorului căminului. Pe pagina dedicată pentru acest lucru sunt listate toate mașinile de spălat într-un tabel, cu nume, uscător asociat și status. În partea stângă a fiecărei mașini se află un buton care afișează toate programările la mașina de spălat din săptămâna curentă.

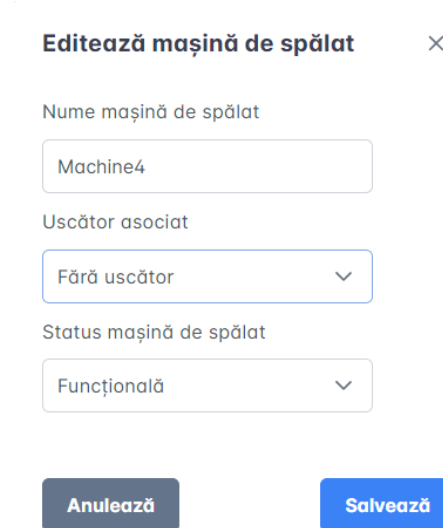


Nume	Uscător asociat	Status
Machine4	Fără uscător asociat	Fără uscător asociat
Machine3	Fără uscător asociat	Fără uscător asociat
Machine1	Dryer1	Disponibil

Data programării	Student
2024-06-15 08:00	taylor.swift@e-uvr.ro
2024-06-13 08:00	taylor.swift@e-uvr.ro
2024-06-15 10:00	lulia.dragoiu2@e-uvr.ro

Figura 4.15: Tabelul cu mașinile din spălătorie

În partea dreaptă a fiecărei mașini se află un buton albastru de editare. Aceasta deschide o fereastră unde administratorul căminului poate să schimbe numele mașinii, uscătorul asociat și statusul. În cazul în care mașina devine defectă și are programări, toate programările vor fi anulate și studenții vor fi anunțați printr-un mail corespunzător.



Editează mașină de spălat

Nume mașină de spălat

Machine4

Uscător asociat

Fără uscător

Status mașină de spălat

Funcțională

Anulează

Salvează

Figura 4.16: Formularul de editare a unei mașini

Pentru administrarea uscătoarelor, administratorul poate să schimbe tipul mașinilor afișate cu ajutorul drop-down-ului din stânga sus. Structura tabelului este aceeași cu cea a mașinilor de spălat. Singura diferență este în cazul în care un uscător devine defect, fiindcă atunci studentul are opțiunea de a păstra programarea la mașina de spălat, fără uscător.

Pentru adăugarea mașinilor, administratorul căminului apasă pe butonul verde ‘+’ din partea de dreapta sus a ecranului. Se deschide o fereastră cu un formular în 3 părți:

- Selectarea tipului mașinii
- Setarea detaliilor mașinii (nume, mașină de spălat/uscător asociat)
- Confirmare

Figura 4.17: Formularul de adăugare mașini - pasul de confirmare

4.3.3 Administrarea studenților

Cea mai importantă parte a aplicației UVTDorms sunt studenții. Pe pagina dedicată administrării studenților dintr-un cămin, administratorul acestuia are la dispoziție un tabel unde sunt listați toți studenții din cămin, într-un tabel cu numele studenților, numărul camerei lor, numărul matricol, și numărul de telefon.

Prenume	Nume	Email	Cămin	Număr cameră	Număr matricol	Număr telefon	Editează	Șterge
Taylor	Swift	taylor.swift@e-uvvt.ro	D13	1	I2345	0765891234		
Emma	Watson	emma.watson@e-uvvt.ro	D13	2	I1234	0763213213		
Iulia	Dragoiu	iulia.dragoiu02@e-uvvt.ro	D13	2	I1239	0747319234		

Figura 4.18: Tabelul cu studenții dintr-un cămin

Administratorul are două butoane pentru fiecare student:

- unul de ștergere, care deschide o fereastră de confirmare
- unul de editare, care deschide o fereastră unde administratorul poate schimba numărul de cameră al studentului

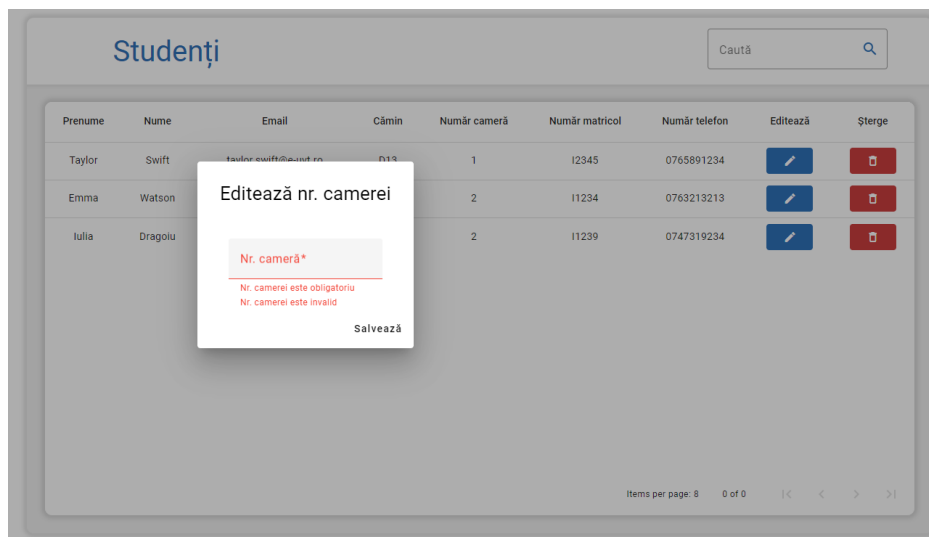


Figura 4.19: Formularul pentru schimbarea numărului de cameră a unui student

Tot pe această pagină sunt listate și cererile de înregistrare la cămin, sub forma unor cărți micuțe cu numele studentului și data în care s-a făcut cererea. Cărțile respective au un buton, care deschide o fereastră cu detaliile cererii.

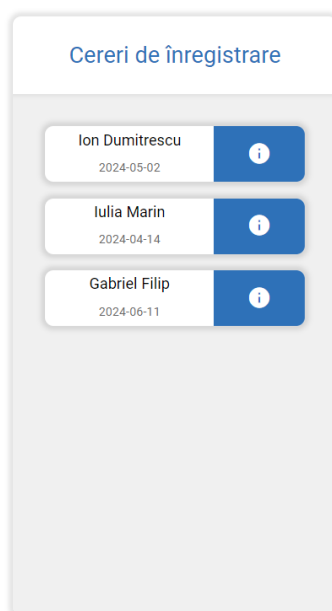


Figura 4.20: Lista de cereri de înregistrare la cămin

Fereastra cu detaliile studentului conține toate informațiile studentului și două butoane:

- butonul ‘*Respinge*’, cu care administratorul căminului refuză cererea făcută de student
- butonul ‘*Acceptă*’, cu care studentului i se confirmă înscrierea în cămin în aplicația UVTDorms

În ambele cazuri, studenții sunt anunțați prin email.

Detalii cerere	
Prenume:	Ion
Nume:	Dumitrescu
Email:	vin.diesel@e-uvt.ro
Cămin:	D13
Camera:	1
Nr. matricol:	I1235
Nr. telefon:	0781123456
Data:	2024-05-02

Respinge
Acceptă

Figura 4.21: Detaliile unei cereri de înregistrare

4.3.4 Administrarea tichetelor de reparații

Pentru gestionarea tichetelor de raportare a problemelor, deschise de studenții căminului, administratorul acestuia are o pagină dedicată. Aici sunt listate toate tichetele din cămin, într-un tabel care conține data creării, titlul și statusul fiecărui tichet.

În stânga fiecărui tichet este un buton care extinde secțiunea și apar detaliile tichetului, cum ar fi tipul de intervenție necesar, descrierea problemei, dacă problema a fost anunțată în trecut, adresa de email a studentului care a făcut tichetul și numărul de cameră.

Pentru a actualiza statusul unui tichet, administratorul apasă pe statusul curent al acestuia și apare o fereastră mică pentru confirmarea schimbării statusului din ‘*Deschis*’ în ‘*Rezolvat*’ și invers.

Cămine				
Evenimente Camere Spălătorie Studenți Tichete				
Tichete				
Data creării 17		Titlu		Status 11
2024-06-05 12:34		Nu se închide ușa		Rezolvat
Tip intervenție	Descriere	Anunțat în trecut 11	Email student 11	Număr cameră
TAMPLAR	Este necesară verificarea și posibila înlocuire a șuruburilor	Nu	lulia.dragoliu02@e-uvr.ro	2
2024-06-14 12:34		Înlocuire bec		Detachable
2024-04-13 12:34		Apo caldă nu funcționează		De

Figura 4.22: Tabelul cu tichetele de reparații

4.3.5 Administrarea evenimentelor

Fiecare administrator de cămin poate organiza evenimente în căminul administrat. Pentru gestionarea evenimentelor, administratorii au o fereastră pe pagina lor de cont. Aici apar toate evenimentele din cămin, cu toate detaliile evenimentului, la fel ca pe pagina de evenimente. Singura diferență reprezintă lipsa butonului ‘Participă’, care este înlocuit de butonul ‘Șterge’. Acesta deschide un mesaj de confirmare a ștergerii evenimentului.


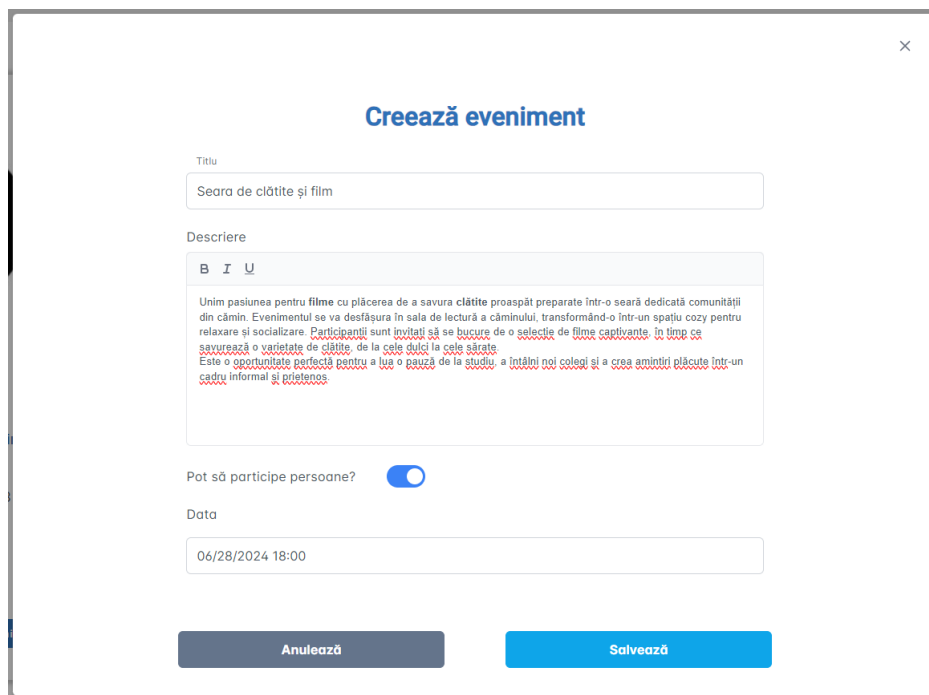
Cămine				
Evenimente Camere Spălătorie Studenți Tichete				
<div> <div>  <p>Tom Hanks</p> <p>Administrator cămin</p> <p>Email: tom.hanks@e-uvr.ro</p> <p>Număr de telefon: 0712345678</p> <p>Cămin: D13</p> <p>Schimbă numărul de telefon</p> <p>Schimbă parola</p> </div> <div> <p>Evenimentele mele</p> <div> <div>+</div> <div> <p>Plant Swap Party: Grow Your Collection!</p> <p>Tom Hanks</p> <p>Calling all plant lovers! Join us for a fun and sustainable Plant Swap Party. Bring a healthy, unwanted plant from your collection and swap it for something new! It's a great way to expand your plant family, declutter your space, and meet other plant enthusiasts. We'll also have resources and tips on plant care available.</p> <p>What to Bring:</p> <ul style="list-style-type: none"> - A healthy, unwanted plant (pots included) - Your enthusiasm for all things green! <p>Delete</p> <p>1 Participant</p> <p>Începe din data 2024/07/04 12:34</p> </div> </div> <div> <div> <p>Volunteer Bake Sale: Supporting the Animal Shelter</p> <p>Tom Hanks</p> <p>Calling all bakers and dessert enthusiasts! We're hosting a Volunteer Bake Sale to raise funds for the local animal shelter. Put your baking skills to the test and donate your delicious creations (or simply come by to indulge!). All proceeds will go towards providing care and comfort for homeless animals.</p> <p>How to Participate:</p> </div> </div> </div> </div>				

Figura 4.23: Tabelul cu evenimentele d.p.d.v al administratorului căminului

Pentru adăugarea evenimentelor noi, administratorul căminului apasă pe butonul verde ‘+’, care deschide o fereastră nouă cu un formular. Aici se introduc titlul evenimentului, descrierea, data și ora în care va avea loc acesta și se specifică dacă studenții își pot arăta intenția de participare la eveniment.



Creează eveniment

Titlu

Seara de clătite și film

Descriere

B I U

Unim pasiunea pentru filme cu plăcerea de a savura clătite proaspăt preparate într-o seară dedicată comunității din cămin. Evenimentul se va desfășura în sala de lectură a căminului, transformând-o într-un spațiu cozy pentru relaxare și socializare. Participanții sunt invitați să se bucure de o selecție de filme captivante. În timp ce savurează o varietate de clătite, de la cele dulci la cele sărate.

Este o oportunitate perfectă pentru a lua o pauză de la studiu, a întâlni noi colegi și a crea amintiri plăcute într-un cadru informal și prietenos.

Pot să participe persoane? ☒

Data

06/28/2024 18:00

Anulează Salvează

Figura 4.24: Formulare de creare de eveniment

4.4 Administrator aplicație

În subsecțiunea următoare este prezentat ghidul de utilizare pentru administratorul aplicației.

4.4.1 Administrarea căminelor

Pentru administrarea căminelor din aplicație, administratorul acestuia are la dispoziție pagina dedicată pentru cămine. Aici căminele sunt listate într-un tabel, cu numele lor, adresa și administratorul. În dreptul fiecărui cămin se află două butoane:

- butonul de ștergere, care deschide o fereastră de confirmare. Căminele nu pot fi șterse dacă au studenți.
- butonul de editare, care permite schimbarea administratorului căminului direct din linia tabelului

Nume cămin	Adresă	Administrator
D13	Street1	Tom Hanks
D14	Street1	
C9	Street2	

Figura 4.25: Tabelul cu căminele

Pentru adăugarea căminelor noi, administratorul aplicației poate apăsa pe butonul verde ‘+’, care deschide o fereastră cu un formular. Aici se pot specifica numele căminului, adresa și, opțional, administratorul.

Adaugă cămin

Nume cămin: C13

Adresă: D13

Administrator: No administrator

Anulează Salvează


Figura 4.26: Formular de adăugare cămin

4.4.2 Gestionarea administratorilor de cămine

Pagina de gestionare a administratorilor de cămine este foarte similară paginii de administrare a căminelor. Aceasta are un tabel în care sunt listați administratorii de cămine, cu detaliile lor, cum ar fi numele, adresa de email, numărul de telefon și căminul administrat.

În dreptul fiecărui administrator se găsesc două butoane:

- butonul de ștergere, care deschide o fereastră de confirmare.
- butonul de editare, care permite schimbarea căminului administrat direct din linia tabelului



Cămine

Administratori cămin

Administratori de cămin

Q Search keyword

Nume	Email	Nr. telefon	Cămin
Tom Hanks	tom.hanks@e-uvt.ro	0712345678	<div><div>D13</div><div><div></div><div></div></div></div>
Hayao Miyazaki	hayao.miyazaki@e-uvt.ro	0782132131	<div><div></div><div><div></div><div></div></div></div>

«

<

1

>

»

Figura 4.27: Tabel cu administratorii de cămine

Pentru adăugarea administratorilor de cămine, administratorul aplicației apasă pe butonul verde ‘+’ care deschide o fereastră nouă cu un formular. Câmpurile formularului permit setarea detaliilor noului administrator de cămin, cum ar fi prenumele, numele, numărul de telefon, adresa de mail și, opțional, căminul asociat.

Noul administrator de cămin primește un email cu parola de conectare și, dacă a fost specificat în formular, căminul la care a fost asociat.

Cămine		Administratori cămin									
<div> <div>Administratori de cămin</div> <table> <tr> <th>Nume</th><th>Email</th></tr> <tr> <td>Tom Hanks</td><td>tom.hanks@e-uvt.ro</td></tr> <tr> <td>Hayao Miyazaki</td><td>hayao.miyazaki@e-uvt.ro</td></tr> </table> </div> <div> <div>Cămin</div> <div>D13</div> <div>✓ ✗ ✎ ✖</div> </div>						Nume	Email	Tom Hanks	tom.hanks@e-uvt.ro	Hayao Miyazaki	hayao.miyazaki@e-uvt.ro
Nume	Email										
Tom Hanks	tom.hanks@e-uvt.ro										
Hayao Miyazaki	hayao.miyazaki@e-uvt.ro										
<div> <div>Adaugă administrator cămin</div> <div> <div>Prenume</div> <div>Andrei</div> </div> <div> <div>Nume</div> <div>Costin</div> </div> <div> <div>Nr. telefon</div> <div>0725007654</div> </div> <div> <div>Email</div> <div>andrei.costin88@gmail.com</div> </div> <div> <div>Cămin</div> <div>D14</div> </div> <div> <div>Anulează</div> <div>Salvează</div> </div> </div>											

Figura 4.28: Formular de adăugare administrator de cămin

Capitolul 5

Concluzii

În subsecțiunile următoare sunt sumarizate detaliile prezentate în lucrare, dar și posibile direcții viitoare.

5.1 Despre aplicația UVTDorms

Aplicația UVTDorms este o aplicație web, accesibilă din browsere și are ca scop digitalizarea unor rutine din cadrul căminelor studențești. Oferă o platformă care aduce o îmbunătățire atât pentru activitățile studenților cât și pentru cea a administratorilor căminelor. Cu ajutorul aplicației studenții pot:

- face programări la spălătorie
- crea tichete de anunțare a problemelor
- fi anunțați de evenimentele din cadrul căminelor
- să se înscrie în alte cămine
- vizualiza programările și tichetele create

Administratorii căminelor pot gestiona toate acestea, pe lângă administrarea camerelor și a studenților care se înscriu în cămine.

Al treilea rol, la fel de important în cadrul aplicației este administratorul acesteia, care adaugă cămine noi și gestionează administratorii acesteia.

Aplicația UVTDorms a fost gândită să fie accesibilă atât de pe desktop, cât și de pe dispozitive mobile. Astfel, implementarea ‘*responsive*’ a aplicației își adaptează interfața la ecranele dispozitivelor de pe care sunt accesate.

5.2 Design aplicație

UVTDorms este o aplicație cu arhitectură modularizată, modernă, care permite o dezvoltare continuă și eficientă. Atât design-ul bazei de date cât și a back-end-ului sunt gândite să fie ușor extensibile. De asemenea, framework-ul Angular, prin natura sa permite extinderea aplicației într-un mod foarte eficient.

Modularizarea componentelor aplicației pe toate straturile sale s-a dovedit a fi foarte utilă, deoarece efectuarea debugging-ului nu a reprezentat niciodată o problemă. Acest lucru se datorează separării responsabilităților dintre modulele aplicației.

5.3 Implementare eficientă cu framework-uri

Aplicația UVTDorms a fost dezvoltată în conformitate cu standardele moderne de dezvoltare software. Am ales să folosesc Angular pe front-end, fiind unul dintre cele mai folosite framework-uri, cu suport oficial și o varietate foarte mare de pachete utile. Pe back-end am folosit framework-ul Spring Boot, care se bazează pe Java, unul dintre cele mai populare limbaje de programare. Spring Boot este unul dintre cele mai întâlnite framework-uri în contextul back-end-ului aplicațiilor web.

Cu ajutorul acestora, implementarea aplicației s-a desfășurat eficient, folosind toate utilitățile framework-urilor utilizate.

5.4 Internaționalizare și ghid asociat

Interfața aplicației UVTDorms a fost gândită să fie una simplă, intuitivă, ușor de învățat și folosit. Paginile aplicației nu sunt prea încărcate și conțin doar elementele necesare.

Fiecare pagină a aplicației are traducere în două limbi, română și engleză, fiindcă studenții Universității de Vest din Timișoara nu sunt doar vorbitori de limba română.

Aplicația are și un ghid de utilizator, pentru toate funcționalitățile aplicației, începând cu cele comune și la final structurate pe roluri. Ghidul are și capturi de ecran, pentru a arăta și exemple pentru utilizatorii care citesc ghidul.

5.5 Direcții viitoare

Având în vedere că aplicația UVTDorms a ajuns la prima versiune finală, aceasta poate să fie instalată pe un server și asociat cu un nume de domeniu și utilizată de studenții și administratorii căminelor. Deși aplicația mai poate fi extinsă cu funcționalități noi, aceasta deja oferă toate funcționalitățile de bază pe care și le-a propus.

Un exemplu bun de extindere a aplicației ar fi integrarea cu procesul de obținere a locurilor în cămine. UVTDorms permite înscrierea studenților în căminele în care au fost deja cazați, dar încă nu suportă procesul de obținere a locurilor.

Bibliografie

- [1] B. Burke. *RESTful Java with Jax-RS*. ” O’Reilly Media, Inc.”, 2009.
- [2] J. D. Drake and J. C. Worsley. *Practical PostgreSQL*. ” O’Reilly Media, Inc.”, 2002.
- [3] N. Gibbins. Cross origin resource sharing. , .
- [4] O. Gierke et al. Spring data jpa-reference documentation. URL <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>. [utoljára megtekintve: 2017. 04. 21.], 2012.
- [5] Java Record vs Final Class. <https://web.archive.org/web/20230519010321/https://www.baeldung.com/java-record-vs-final-class>.
- [6] M. Jones, J. Bradley, and N. Sakimura. Json web token (jwt). Technical report, IETF, 2015.
- [7] R. C. Martin. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.
- [8] A. Pantaleev and A. Rountev. Identifying data transfer objects in ejb applications. In *Fifth International Workshop on Dynamic Analysis (WODA’07)*, pages 5–5. IEEE, 2007.
- [9] C. Scarioni and M. Nardone. *Pro spring security: securing spring framework 5 and boot 2-based Java applications*. Apress, 2019.
- [10] L. Spilca. *Spring security in action*. Simon and Schuster, 2020.
- [11] B. Varanasi and S. Belida. *Spring Rest*. Apress, 2015.