UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTAȚIE

Proiectare Software

Final Assignment

Photoghaphy Album Site Application

Boloș Andrei Nicolae
GRUPA: 30236

CUPRINS

1. *Project Specification*

Obiectivul principal al aplicaţiei reprezintă proiectarea şi implementarea unui site pentru Afişarea unui album de imagini ale unor utilizatori, după ce aceştia s-au conectat cu success, într-o aplicaţie Web care se bazează pe Java.

2. *Functional Requirements*

În funcţie de *user*, atributul de admin se setează din back-end pe *true* doar la persoanele autorizate cu access la codul sursă.
Utilizatorul normal are disponibile următoarele operaţii:

- ➔ Operaţii pe baza de date pentru ***User*** – CRUD;
- ➔ Operaţii pe baza de date pentru ***Photo*** – CRUD;
- ➔ Actiunea de a cumpăra o poză;
- ➔ Trimitere mail utilizatorului a cărui poză se intenţionează a fi cumpărată;
- ➔ Trimitere mail de confirmare a comenzii utilizatorului a care are intenţia de a cumpăra o poză;
- ➔ Ştergerea unui *user* normal de către un *admin*.

3. *Use Case Model*

Use-Case:
Level:
Primary Actor:
Main success scenario:
Extensions:

Use-Case1: Register
Level: User-Goal
Primary Actor: User
Main success scenario: Successful register
Extensions: Format ilegal de date

Use-Case2: LogIn
Level: Subfunction
Primary Actor: User
Main success scenario: Successful login

Use-Case3: LogOut
Level: Subfunction
Primary Actor: User

Main success scenario: Successful logout

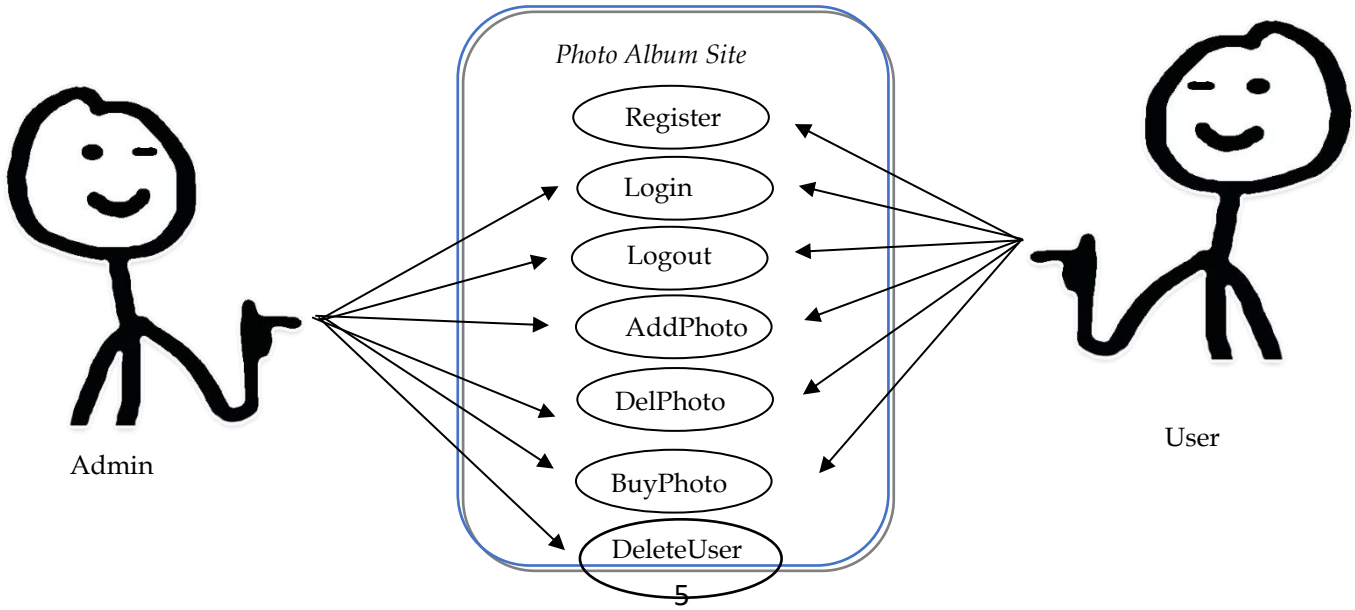
Use-Case4: Delete/Add a Photo
Level: User
Primary Actor: User
Main success scenario: Userul apasă butoane care adaugă poze sau le sterge din baza de date. Se selecteaza poza din File Exploprer

Use-Case5: Buy a Photo
Level: Client-Goal
Primary Actor: User
Main success scenario: Un user se uita in lista de poze ale altui user, dă hover, se afiseaza pretul si optiunea de a Cumpara
Extensions: clientul nu are sufficient monetar in cont, rezulta in mesaj de eroare.

Use-Case6: Delete a user
Level: admin
Primary Actor: admin
Main success scenario:Adminul poate sterge din baza de date. Se selecteaza userul dintr-o lista de useri.

Diagrama UML (Unified Modeling Language) reprezintă structura logică a proiectului. Specifică împărţirea în pachete, clase şi relaţii.
Ea poate conţine şi atributele, constructorii şi metodele claselor.

Diagrama Usecase:



4. Supplementary Specification

Non-Functional Requirements:

- Portabilitate – poate functiona pe diferite device-uri, cu diferite rezolutii;
- Usability: user friendly interface easily navigable
- Security:password encryption pt a face dificil furtul de date;
- Internationaliazre si localizare: website ul poate fi tradus in mai multe limbi;

Design Constraints:

Partea de backend a proiectului o sa fie conceputa cu ajutorul limbajului Java si Spring, parte de frontent o sa fie facuta in React iar Baza de date o sa fie implementata si stocata cu ajutorul aplicatiei MySQL, simplificarea codului va fi realziata cu ajutorul Lombok, maparea intre baza de date si java se face cu JPA.

IDE-ul folosit ieste InteliJ. In cadrul proiectului o sa fie folosita o arhitectura stratificata care o sa respecte principiile SOLID.

O sa fie cinci mari module: Model, Repo, UI, Controller si Service, care vor avea fiecare multiple clase.

5. Domain Model

Modelele principale ale proiectului sunt *User*-ul şi *Photo*-ul.

Un User este văzut ca o persoană care poate deţine nişte poze, iar poza este obiectul principal al acestei expoziţii.

Această afirmaţie se poate traduce în implementarea aplicaţiei ca fiind necesitatea a două entităţi: ***Photo***, ***User***.

O ***Poză*** poate avea următoarele atribute:

- *id* – un identificator unic al pozei;
- *name* – un nume sugestiv;
- *description* – o descriere sugestivă;
- *path* – calea de referinţă către zona de memorie unde este stocată poza;
- *user* – poate fi atribuit si un deţinător al pozei.

Un ***User*** poate avea următoarele atribute:

- id – un identificator unic al userului;
- name – un nume sugestiv;
- email – un email cu care se loghează;
- password – o parolă cu care se loghează;

- admin – determină dacă acest user are drepturi de *admin* sau nu;
- photoList – o listă cu pozele pe care acest user le deține.

6. *Architectural Desing*

- *Conceptual Architecture*

Arhitectura are la bază structura pe nivele – Layered Architecture.

a. Stratul de prezentare **(Presentation Layer)**

Acest strat reprezintă partea vizibilă de către utilizator şi are rolul de a prelua datele de intrare din interfaţa grafică şi de a le trimite mai departe către următorul nivel.

Pentru acest nivel s-au folosit tehnologii precum HTML, CSS, Angular/React.

Această aplicaţie poate să ruleze pe desktop sau pe web, dar este implementată doar rularea într-un browser.

b. Stratul de acces la date **(Data Access Layer)**

Acest strat are rolul de a prelua datele din interfaţa grafică şi le stochează în baza de date şi le trimite mai departe pentru prelucrare ulterioară. În baza de date sunt stocate modelele aplicaţiei.

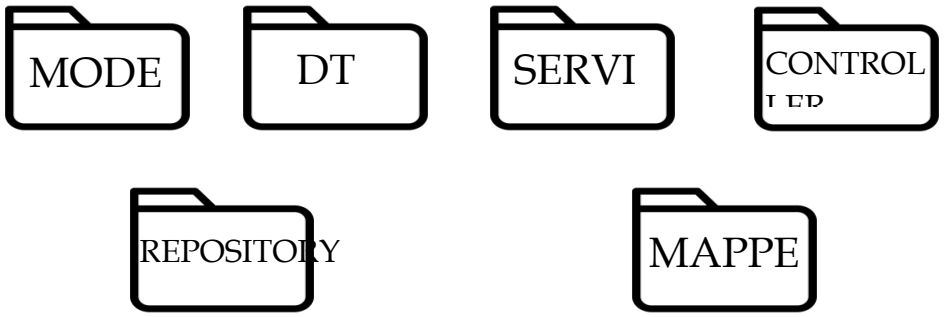
c. Stratul de logică **(Business Layer)**

Acest strat se ocupă cu prelucrarea datelor ce sunt stocate în baza de date şi trimiterea lor spre afişare.

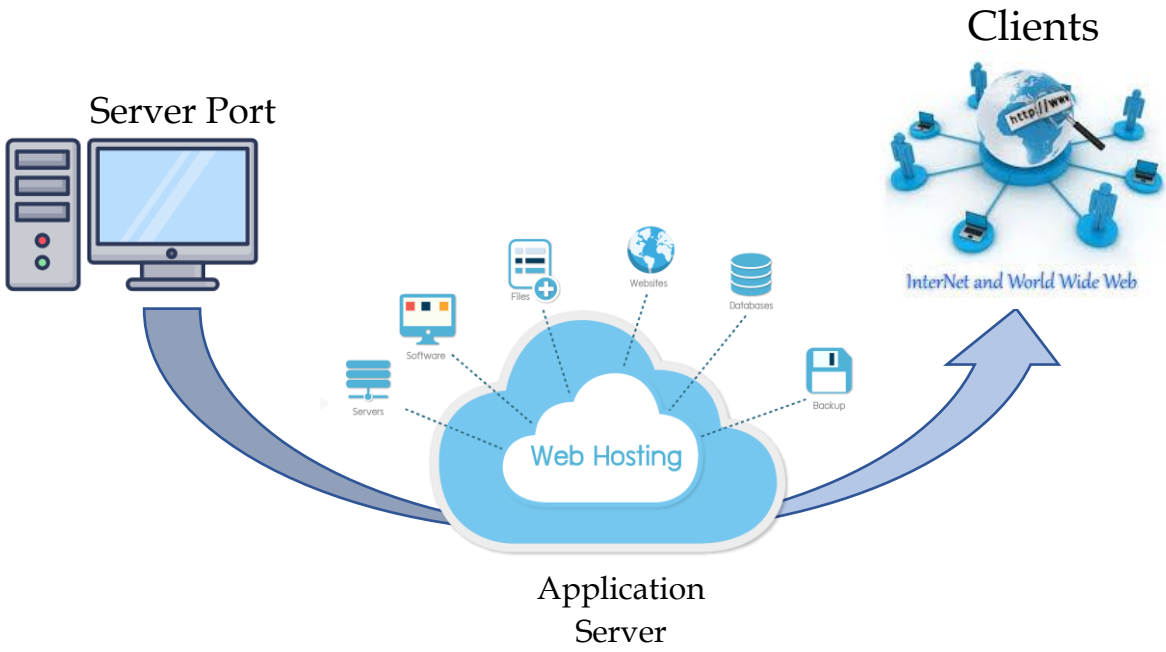
Aici se află detaliile legate de implementare, precum implementarea funcţiilor de acces la baza de date, de modificare a proprietăţilor fiecărui obiect şi diferite sarcini.

Arhitectura se poate adapta după cerinţe proaspăt primite sau actualizate.

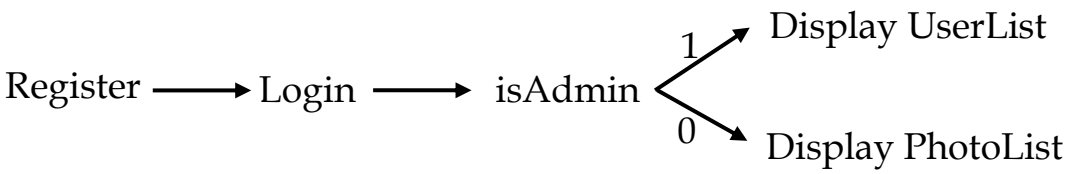
- *Package Design*



- *Component & Deployment Diagrams*



7. Design Model



- ***Scenario 1:***

Dacă un user nu este inregistrat in baza de date, atunci se poate inregistra cu o adresa de email, un nume si o parola. Parola este encriptata.

Odata inregistrat in baza de date, acest user poate sa dea Login folosind email-ul si parola introduse la inregistrare.

Dacă acest user are statutul de Admin, atunci el poate vedea o lista cu toti userii existenti in baza de date.

- ***Scenario 2:***

Dacă un user nu este inregistrat in baza de date, atunci se poate inregistra cu o adresa de email, un nume si o parola. Parola este encriptata.

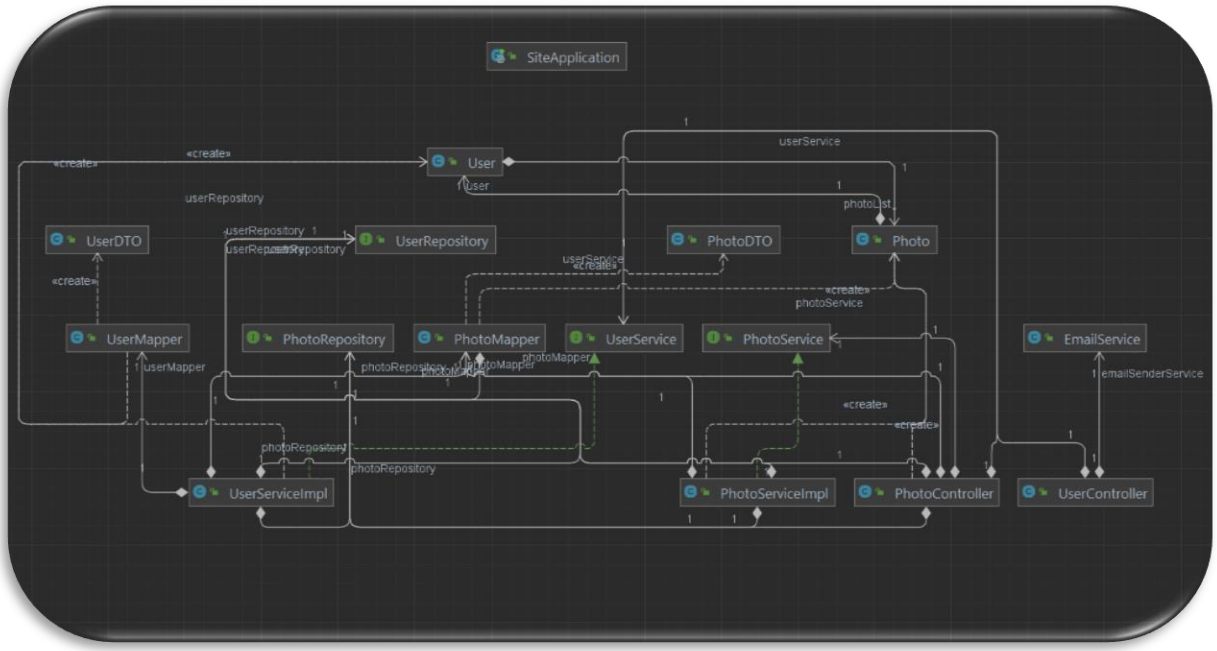
Odata inregistrat in baza de date, acest user poate sa dea Login folosind email-ul si parola introduse la inregistrare.

Dacă acest user nu are statutul de Admin, atunci el poate vedea o lista cu pozele stocate in baza de date.

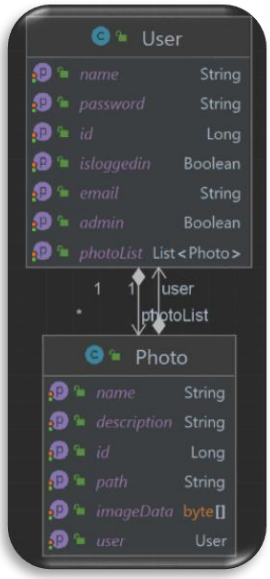
Aceste poze se pot modifica, sterge sau se pot afisa detaliat.

Lista se poate sorta crescator sau descrescator in functie de ID, sau se poate filtra pentru a permite doar afisarea pozelor ce apartin unui anumit user.

8. Class Diagram: UML



9. Data Model



Data model-ul aplicatiei acestea este unul relational. Se foloseste o baza de date ca sa se stocheze entitatile principale: User si Photo.

Relatia dintre User si Photo este OneToMany, deoarece un user poate avea o lista de poze.

10. System Testing

Partea de testare reprezinta validarea corectitudinii functionalitatilor si a implementarilor acestora.

Exista testare automata a sistemului, fiind implementate teste unitare ce verifica functionalitatea operatiilor pe baza de date: Create, Read, Update, Delete.

Majoritatea testării s-a facut verificand comportamentul aplicatiei live, adica la fiecare actualizare s-a verificat aspectul vizual, corectitudinea rezultatelor si solutionarea erorilor care apăreau mai mult sau mai putin neprevazut.

Testarea manuala a reprezentat baza verificarii faptului ca aplicatia merge sau nu cum a fost intentionat a functiona.

11. *Future Improvements*

Pentru ca aplicatia sa poata fi folosita la urmatorul nivel, eventual acesta ar putea fi chiar un nivel de productie, aceasta trebuie sa fie pusă la punct din mai multe puncte de vedere:

- Securitate;
- Accesibilitate & Uşurinţă de folosire;
- Aspect vizual;
- Utilitate;
- Noi functionalitati, precum posibilitatea de tranzactii intre useri.

12. *Conclusion*

Acest proiect a avut un rol important in dezvoltarea mea pe plan profesional, deoarece am impresia că acest tip de proiecte fac parte din probabil cea mai utilizată ramură din domeniul IT.

As fi dorit sa depun mai mult timp acestui proiect, incat sa poata fi utilizat mai mult decat cu scop educativ.

13. *Bibliografie*

Link Git:

Link Resurse:

https://www.javaguides.net/2019/07/spring-data-jpa-save-findbyid-findall-deletebyid-example.html

https://www.youtube.com/watch?v=tLBX9fq813c&list=PLGRDMO4rOGcNzi3CpBWsCdQSzbjdWWy-f&index=1

inspiratie efecte: https://www.soaringcloudsrecords.com

https://www.javainuse.com/fullstack/imageupload

https://medium.com/shoutloudz/spring-boot-upload-and-download-images-using-jpa-b1c9ef174dc0

tutorial how to upload an image:

https://www.javainuse.com/fullstack/imageupload

page up scroll button:

https://www.w3schools.com/howto/howto_js_scroll_to_top.asp

websocket:
https://www.youtube.com/watch?v=A3zBohNgHKE
Export HTML table to PDF
https://www.youtube.com/watch?v=Kik1SvebqTg
Export XML
https://www.youtube.com/watch?v=koYZ_xS8mf0
Encriptare parola
https://www.youtube.com/watch?v=a3ZdcDvUwdY