**Logo

Description automatically generated**

**DOCUMENTAȚIE**

Distributed Systems

**Assignment 1**

***Request-Reply***

***Communication***

Boloș Andrei Nicolae

GRUPA: 30244

***CUPRINS***

[***CUPRINS*** 2](#_Toc150335537)

[***1.*** ***Project Specification*** 3](#_Toc150335538)

[***2.*** ***Functional Requirements*** 3](#_Toc150335539)

[***3.*** ***Supplementary Specification*** 3](#_Toc150335540)

[***4.*** ***Domain Model*** 4](#_Toc150335541)

[***5.*** ***Architectural Desing*** 4](#_Toc150335542)

[***6.*** ***Conceptual Architecture Diagram*** 5](#_Toc150335543)

[***7.*** ***Deployment Diagram*** 6](#_Toc150335544)

[***8.*** ***System Testing*** 7](#_Toc150335545)

[***9.*** ***Future Improvements*** 7](#_Toc150335546)

[***10.*** ***Conclusion*** 7](#_Toc150335547)

[***11.*** ***Bibliografie*** 7](#_Toc150335548)

1. ***Project Specification***

Obiectivul principal al aplicației reprezintă proiectarea și implementarea unui site pentru afișarea listei de utilizatori sau a listei dedispozitive, după ce aceștia s-au conectat cu success, într-o aplicație Web, Restful, care se bazează pe Java și React.

1. ***Functional Requirements***

În funcție de *user*, atributul de *role* este implicit “*client*”, dar un *admin* poate modifica acest rol.

Utilizatorul are disponibile următoarele operații:

* Operații pe baza de date pentru ***User*** – CRUD, dacă are rolul de “*admin*”;
* Operații pe baza de date pentru ***Device*** – CRUD;

1. ***Supplementary Specification***

*Non-Functional Requirements:*

* Microservicii – se folosesc microservicii, cate unul pentru User & Device;
* Securitate Simpla – pe baza numelui, se afiseaza pagina corespunzatoare rolului de admin/client, fara a fi permisa schimbarea path-ului catre cealalta pagina.

*Design Constraints:*

Partea de backend a proiectului este 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 Workbench, 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, Repository, DTO, Controller si Service, care vor avea fiecare multiple clase.

1. ***Domain Model***

Modelele principale ale proiectului sunt *User*-ul și *Device*-ul.

Un ***User*** este văzut ca o persoană care poate deține niște dispozitive, iar dispozitivul este obiectul principal al acestei expoziții.

Această afirmație se poate traduce în implementarea aplicației ca fiind necesitatea a două entități: ***Device***, ***User***.

Un ***Device*** poate avea următoarele atribute:

* *id* – un identificator unic;
* *description* – o descriere sugestivă;
* *address* – adresa la care se afla dispozitivul;
* *maximumHourlyEnergyConsumption* – numar reprezentand consumul maxim al dispozitivului intr-un interval de o ora.
* *userId* – id-ul userului care il deține.

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

• *id* – un identificator unic al userului;

• *name* – un nume sugestiv;

• *role* – determină dacă acest user are drepturi de *admin* sau nu;

1. ***Architectural Desing***

* *Conceptual Architecture*

Arhitectura are la bază structura pe nivele – Layered Architecture.

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

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

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

Această aplicație implementată pentru rularea într-un browser.

* 1. Stratul de acces la date **(Data Layer)**

Acest strat are rolul de a stoca datele din interfața grafică în baza de date și le trimite mai departe pentru prelucrare ulterioară. În baza de date sunt stocate modelele aplicației. Aici se utilizeaza baze de date relationale prin aplicatia MySQL.

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

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

Aici se află detaliile legate de implementare, precum implemen-tarea funcțiilor de acces la baza de date, de modificare a proprietăților fiecărui obiect și diferite sarcini. Aici se afla cele doua microservicii realizate cu ajutorul framework-ului Spring.

Arhitectura se poate adapta după cerințe proaspăt primite sau actualizate.

1. ***Conceptual Architecture Diagram***

A diagram of a system

Description automatically generated

1. ***Deployment Diagram***

A diagram of a diagram of a diagram

Description automatically generated

1. ***System Testing***

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

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.

1. ***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 & Ușurință de folosire;
* Aspect vizual;
* Utilitate;

1. ***Conclusion***

Acest proiect a avut un rol important in dezvoltarea mea pe plan profesional.

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

1. ***Bibliografie***

Link Git:

Link Resurse:

<https://dsrl.eu/courses/sd/>