Titlul lucrării

PROIECT DE DIPLOMĂ

Autor: **Prenume NUME**

Conducător științific: **Titlu. ing. Prenume NUME**

|  |  |
| --- | --- |
| DECAN  **Prof. dr. ing. Liviu MICLEA** | Vizat,  DIRECTOR DEPARTAMENT AUTOMATICĂ  **Prof. dr. ing. Honoriu VĂLEAN** |

Autor: **Prenume NUME**

Titlul lucrării

1. **Enunțul temei:** *O scurtă descriere a temei proiectului de diplomă*
2. **Conținutul proiectului:** *(enumerarea părților componente) Pagina de prezentare, Declarație privind autenticitatea proiectului, Sinteza proiectului, Cuprins, Titlul capitolului 1, Titlul capitolului 2,… Titlul capitolului n, Bibliografie, Anexe.*
3. **Locul documentării:** *Universitatea Tehnică din Cluj-Napoca, alte locuri dacă este cazul*
4. **Consultanți:** *ing. Prenume Nume (dacă este cazul)*
5. **Data emiterii temei:**
6. **Data predării:**

Semnătura autorului

Semnătura conducătorului științific

**Declaraţie pe proprie răspundere privind**

**autenticitatea proiectului de diplomă**

Subsemnatul(a) **Prenume NUME**  , legitimat(ă) cu CI/BI seria nr. , CNP ,

autorul lucrării:

elaborată în vederea susținerii examenului de finalizare a studiilor de licență la **Facultatea de Automatică și Calculatoare**, specializareaChoose an item.**,** din cadrul Universității Tehnice din Cluj-Napoca, sesiunea Choose an item. a anului universitar 2022-2023, declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, pe baza cercetărilor mele și pe baza informațiilor obținute din surse care au fost citate, în textul lucrării, și în bibliografie.

Declar, că această lucrare nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române și a convențiilor internaționale privind drepturile de autor.

Declar, de asemenea, că această lucrare nu a mai fost prezentată în fața unei alte comisii de examen de licență.

În cazul constatării ulterioare a unor declarații false, voi suporta sancțiunile administrative, respectiv, *anularea examenului de licență*.

Data Prenume NUME

(semnătura)

**SINTEZA**

proiectului de diplomă cu titlul:

Titlul lucrării

Autor: **Prenume NUME**

Conducător științific: **Titlu. ing. Prenume NUME**

1. Cerințele temei:

2. Soluții alese:

3. Rezultate obținute:

4. Testări și verificări:

5. Contribuții personale:

6. Surse de documentare:

Semnătura autorului

Semnătura conducătorului științific

Cuprins

[1 Introducere 2](#_Toc477457095)

[1.1 Context general 2](#_Toc477457096)

[1.2 Obiective 2](#_Toc477457097)

[1.3 Specificații 2](#_Toc477457098)

[2 Studiu bibliografic 3](#_Toc477457099)

[3 Analiză, proiectare, implementare 4](#_Toc477457100)

[4 Concluzii 5](#_Toc477457101)

[4.1 Rezultate obținute 5](#_Toc477457102)

[4.2 Direcții de dezvoltare 5](#_Toc477457103)

[5 Reguli de formatare 6](#_Toc477457104)

[5.1 Formatarea paginii 6](#_Toc477457105)

[5.2 Titluri și stiluri 6](#_Toc477457106)

[5.3 Figuri, tabele și ecuații 7](#_Toc477457107)

[5.3.1 Figuri 7](#_Toc477457108)

[5.4 Tabele 7](#_Toc477457109)

[5.5 Ecuații 7](#_Toc477457110)

[5.6 Referințe bibliografice 8](#_Toc477457111)

[6 Bibliografie 9](#_Toc477457112)

# Introducere

## Context general

În introducere familiarizați cititorul cu motivația lucrării, plasați lucrarea într-un context care să permită cititorului să înțeleagă obiectivele.

Descrieți importanța lucrării, de ce merita să o faceți, plasați ideile într-un context larg.

Susțineți studiul: de ce exact aceasta aplicație/implementare. Comentați asupra aspectelor teoretice sau practice care v-au făcut să o alegeți.

În funcție de natura lucrării, ar putea fi necesar să prezentați informații de fond asupra domeniului în care se încadrează aplicația, mai ales dacă aveți o lucrare într-un domeniu multidisciplinar. În acest caz, puteți introduce terminologia pe care o utilizați în continuare.

Descrieți pe scurt lucrarea: ce conține fiecare capitol.

## Obiective

Enumerați și explicați obiectivele lucrării: ce v-ați propus să realizați în contextul prezentat anterior.

Obiectivele pot fi prezentate sub formă de listă care să evidențieze precis orientarea lucrării, să identifice conceptele fundamentale pe care le studiați, să stabilească scopul aplicației pe care o realizați, sau enunțați întrebările la care intenționați să răspundeți în lucrare.

## Specificații

În specificațiile lucrării detaliați cerințele. Descrieți ce intenționați să obțineți. Vă puteți referi la funcțiile aplicației, interfață, nivele de performanță, structuri de date, elemente, securitate, fiabilitate, calitate, limitări, etc.

# Studiu bibliografic

2.2 MVC Framework

MVC este acronimul de la Model View Controller, acesta fiind un șablon de arhitectură pentru o aplicație software. Acesta oferă un mod de a structura modul prin care o aplicație este construită, separând aceasta în trei componente separate: Model, View și Controller. Fiecare din aceste componente este destinată pentru un aspect specific al aplicației.

Aici pune o diagrama cu MVC

2.2.1 Model

Model este componenta care se ocupă cu logica legată de datele/informațile? cu care utilizatorul poate să lucreze. Această componentă ține toată informația incapsulată pentru a putea fii acesată doar prin intermediul unor metode speciale care se ocupă cu manipularea acestor informații, cum ar fii obținerea informaților din baza de date sau validarea acestora.

2.2.2 View

Componenta View este reprezentată de interfața utilizatorului. Aceasta definește modul in care informația de la componenta model este reprezentată pentru utilizatorul aplicației. O astfel de componentă de obicei este alcătuita din elemente de HTML, CSS si JavaScript conținând butoane, dropdown-uri și check box-uri cu care utilizatorul poate interacționa.

2.2.3 Controller

Rolul componentei Controller este de a asigura un canal intre componenta Model și View.Aceasta componentă există pentru a procesa inputul primit de la utlilizator după care sa fie procesat si transmis componentei Model in funcție de request-ul primit. Cu ajutorul Controller-ului se asigură separarea si independența fiecaărei componente. Controllerul este componenta care decide modul în care informatia este folosită pentru întreaga aplicație folosinduse de View pentru a afișa aceste informații si de Model pentru a le obține din baza de date.

2.2.4 Avantajele MVC/ sa scriu asta la 2.2 MVC

Avantajul major penntru acest framework este că oferă o spearare clară pentru fiecare componentă a aplicației astfel programatorii pot sa lucreze independent pentru diferite aspecte, schimbari ale unei componente nu afeactează in mod direct restul componentelor doar dacă aceste schimbari urmaresc șablonul și regulile arhitecturii MVC. Acest framework oferă posibilitatea de a reutiliza codul, modularitatea acestuia.

Compromisul pentru aceste avantaje este creșterea complexitații programului fiind împărțit in diferite componente.

2.3 Spring Boot

Spring Boot este un framework bazat pe limbajul de programare Java, deschis pentru publicaul larg/”open source”, care simplifică dezvoltarea aplicaților Java în special pentru aplicatile web și miroservicii. Este construit cu ajutorul framework-ului Spring, oferind un mod de a configura și a lansa aplicații Java care pot să ajungă direct în stadiul de producție.

Principalul scop al acestui framework este de a minimiza procesul nesesar de configurare și înființare pentru o aplicație Spring prin adoptarea unui set de configurări logice implicite permițând astfel programatorilor să înceapă dezvoltarea unei aplicații noi fară fara a aloca prea mult timp pe o schemă de amploare.

Unele din caracteristicile cheie ale acestui framework este:

* Auto-Configurarea: Spring Boot poate să configurează mai multe componente în funcție de dependințele din proprietațile aplicației, astfel se elimină nevoia configurării manuale și se reduce probabilitatea apariției unor erori
* Server Încorporat: Spring Boot are în componență un server încorporat cum ar fii Tomcat care oferă posibilitatea aplicaților sa fie pornite fară a fi nevoie de instalarea pe un server extern
* Managementul Dependențelor: Spring Boot simplifică managementul versiunilor dependințelor astfel se asigura o buna compatibilitate între acestea
* Integrarea ecosistemului Spring: Spring Boot integrează alte componente din ecosistemul Sping cum ar fi: Spring Security, Spring Cloud și Spring Data. Toate acestea fiind integrate intr-un ecosistem se pot folosii necesitând un nivel minim de configurare.

2.3.1 Dependințele în Spring Boot

În Spring Boot dependințele sunt genstionate de instrumente de automatizare cum ar fii: Apache Maven sau Gradle. Aceste dependințe se definec în aplicație in fișierul “pom.xml” in Maven sau în “build.gradle” în Gradle. Cele mai folosite dependințe pentru o aplicatie web construită cu Spring Boot sunt:

* Spring-Boot-Starter-Web: această dependtință include librariile necesare pentru o aplicație web, cum ar fi un server încorporat (Tomcat) și librarii pentru gestionarea cererilor și raspunsurilor HTTP
* Spring-Boot-Starter-Data-Jpa: această dependință oferă suport pentru folosirea Java Persistence API cu Spring Data, incluzând librariile necesare pentru legătura cu baze de date relaționale.
* MySql-Connector-Java: este o dependință specifică bazei de date relaționale MySql. Include driver-ul JDBC care oferă posibilitatea aplicației construită cu Spring Boot să se conecteze la baza de date MySql

2.4 Spring Security

Spring Security este un framework folosit pentru aplicațile Java create cu ecosistemul Spring, oferind un mod de a securiza aplicația. Acest framework pune la dispoziția programatorului mai multe caracteristici si opțiuni pentru a manipula autentificarea, autorizarea și alte nevoi legate de securitate pentru aplicație, intr-un mod fexibil și modular. Unele din obiectivele pe care Spring Security le îndeplinește sunt urmatoarele:

* Autentificarea: acest framework pune la dispoziție diferite mecanisme pentru autentificarea utilizatorilor cum ar fi autentificarea bazată pe un formular, autentificarea simpla HTTP sau cu ajutorul token-urilor JWT(Json Web Token)
* Autorizarea: Spring Security oferă mecanisme pentru a controla accesul utilizatorilor pentru diferite părți ale aplicației și poate securiza URL-uri sau anumite metode pentru a fi accesate din exteriorul aplicației. Aceste mecanisme pot să fie bazate pe rolurile utilizatorilor, permisiuni sau reguli predefinite.

2.4 HTML CSS JavaScript

Separarea acestor limaje

2.5 HTML

Acronimul HTML vine de la “Hypertext Markup Language”, este un limbaj standard folosit pentru creearea de pagini și aplicații web. Structura și conținutul acestuia se reprezintă îm WWW(World Wide Web). Acest limbaj se foloseste de o serie de adnotații pentru a creea și pentru a structura o pagina web.

Codul pentru un document HTML se poate scrie în orice editor de text, singura condiție fiind ca extensia fișierului sa fie “.html”. Orice document HTML contine anotația: “<!DOCTYPE html>” care reprezintă locul de unde incepe codul HTML. Structura unei astfel de pagini se reprezintă prin anotațile „<html> </html>” care incadrează codul prorpiu-zis, „<head> </head>” care conține titlul paginii sau informații despre formatare, „<body> </body>” unde se reprezintă informația din pagină cum ar fi paragrafe cu text sau imagini, uneori din structura unei pagini web face parte și „<footer> </footer>”, reprezentând partea de jos a paginii unde se afla în general informații de contact de exemplu numarul de telefon sau adresa de e-mail.

2.6 CSS

CSS (Cascading Style Sheets) este un limaj folosit împreuna cu limbajul HTML pentru a îmbunatății aspectul vizual pentru o pagina web. Cu ajutorul acestui limbaj programatorul poate să adauge sau să schimbe aspectele vizuale ale paginii după proprile preferințe, cum ar fi: font-urile, mărimea literelor, culorile textului, culorile de fundal sau redimensionarea imaginilor.

Pentru a stiliza un element specific din HTML, CSS pune la dispoziție o serie de selectori pentru componentele HTML. Acești selectori se pot baza pe numle elementelor (h1{}), id-urile acestora (#id{}) sau pe relațile dintre elemente(body > p {}).

Acest limbaj pune la dispozitie si proprietăți prin care elementele se pot poziționa sau alinia intr-o pagină cum ar fi: “display” , “flexbox” sau “grid” . Cu aceste proprietăți se pot obține pagini care își schimbă aspectul în funcție de mărimea ecranului, aceeași aplicație poate să aibă un aspect diferit pe un Desktop față de aspectul de pe un telefon mobil.

2.6.1 Bootstrap

Bootstrap este un framework destinat pentru partea clientului adică pentru interfață. Acesta dispune de o serie de colecții de componente HTML, CSS si JavaSpric predefinite astfel procesul de dezoltare a unei pagini sau aplicații web este mult mai rapid și mai simplu de realizat.

Unul din marile avantaje pentru folosirea acestui framework este că majoritatea componentelor predefinite oferă un design? care este adaptabil pentru diferite dispozitive. Bootstrap oferă si un sistem în care este împărțit ecranul sub forma de table cu 12/doisprezece coloane care se poate personaliza creeând diferite aspecte pentru simplificarea poziționării componentelor într-o pagină.

2.7 JavaScript

JavaScript este un limaj de programare de nivel înalt, orientat pe obiect care funcționează in browserele web. În cele mai multe cazuri este folosit pentru dezvoltarea aplicaților sau paginilor web. Cu acest limbaj se adaugă posibilitatea de a interacționa dinamic cu elementele create cu limajul HTML în pagină. Acest limaj se ocupă în principal cu gestionarea evenimentelor care apar într-o interfață web cum ar fi: click-uri, mișcări ale mouse-ului sau interacțiuni cu tastatura. Gestionarea evenimentelor oferă posibilitatea actualizării paginilor web fară a fi nevoie de reîncărcarea paginii.

Chiar daca la început a fost dezvoltat pentru creearea de cod pentru interfața aplicaților prin intermediul framework-ului Node.js acest limaj se poate utiliza și pentru construirea serverului, astfel acest limaj oferă suport pentru întreaga aplicație web.

Toate browserele din ziua de azi oferă suport și motoare pentru executarea codului JavaScript în mod eficient. În general aplicația trebuie să se muleze pentru nevoile utilizaorilor de aceea find important acest aspect în care se poate folosii orice browser după preferințe, JavaScript fiind cea mai bună alegere pentru dezvoltarea interfețelor aplicaților web.

JavaScript pune la dispoziție mecanisme pentru gestionarea informației primite de la un server pentru o interfață dintr-o pagina web. Fiind un limaj orientat pe obiecte suporta tipul de date obiect dar are si tipuri de date simple cum ar fi: “string” în care se pot stoca litere, cuvinte și numere, “numbers” unde se pot stoca doar numere și “arrays” unde se pot stoca diferite tipuri de date sub formă de șiruri unidimensionale sau bidimensionale. JavaScript poate sa aducă informație în pagina web din surse externe cum ar fi API-urile (Application Programming Interfaces) și astfel se poate afișa pe pagina conținut dinamic prin intermediul actualizărilor în timp real.

S-au dezvoltat mai multe framework-uri pentru o dezvoltare mai eficientă si mai rapidă a interfțelor web, toate acestea având la bază limbajul JavaScript, unele dintre cele mai populare framework-uri care se ocupă cu asa ceva sunt: React, Angular și jQuery.

2.7 React

React este una dintre cele mai populare framework-uri pentru dezvoltarea interfețelor destinate utilizatorilor și a aplicaților web, arhitectura pe care este bazată se folosește de reutilizarea codului și modularitatea acestuia.

Proprietatea de modularitate este atinsă prin divizarea interfețelor în componente independent construite care conțin propria logică de gestionarea a datelor, propriul mod de aranjare și stilizare a elementelor în pagina, chiar și propriul set de teste. Prin această modularizare se pot obține mult mai ușor aplicați complexe în care este nevoie de o mulțime de elemente la care se poate lucra independent și în paralel aducând un plus de eficientizare a procesului si de îmbunatățire a vitezei de lucru.

React introduce o reprezentare virtuală pentru DOM(Document Object Model) care se numește “Virtual DOM”. În loc să se folosească DOM-ul propriu-zis, React construiește o reprezentare virtuală a acestuia în memorie care îmbunătățește performanța programului, tot procesul realizânduse mult mai rapid deoarece în momentul în care sunt schimbări în starea sau în proprietățile unei componente, React actualizează efficient doar părțile necesare din DOM și utlizează la un minim necesar acest proces.

Framework-ul React folosește o sintaxă declarative față de una imperativă.Prin sintaxa declarative programatorul scrie codul astfel încât să descriei aplicației modul în care vrea sa fie afișată informația primită iar React se ocupă de actualizarea interfeței pentru a arata schimbarile în funcție de ce informații primește, astfel procesul de programare a interfeței se simplifică nemaifiind nevoie ca programatorul sa se ocupe manual de fiecare component DOM pentru diferite informații primite ca intrare pentru componentă.

Față de o aplicație web clasică care nu folosește un framework pentru creearea interfeței unde in mod normal pentru a creea componente care se pot afișa pe ecran se folosește HTML, în React se utilizează JSX ( JavaScriptXML ) care este o extensie de sintaxă unde se încorporează cod asemănător cu cel HTML în JavaScript. Extensia JSX este formată dintr-o combinație de structurare a codului HTML cu logica din spatele limajului JavaScript. În acest mod se pot definii toate structurile de componente și proprietățile deja existente în HTML dar intr-un mod intuitiv și mai ușor de recitit pentru alți programatori.

În acest framework se introduce conceptul de “React Hooks”. Această caracteristică pune la dispoziția programatorilor opțiunea de a folosii starea de utilizare a unei componente și alte proprietăți ale framework-ului fără a mai fi nevoie de componente Clasă. Există mai multe tipuri de “Hooks”, dar cele folosite în majoritatea aplicaților sunt “useState”, “useEffect” și “useContext” care simplifică managementul stării componentelor, maipularea evenimentelor secundare și împărțirea contextului între componente. De exemplu pentru a stoca și a seta starea unei componente se folosește useState, aceast “Hook” se declară astfel: const [componentă, setComponentă] = useState(‘’); astfel starea componentei este stocată in constanta numită “componentă” și se poate seta cu funcția “setComponentă”, valoarea de bază a acesteia fiind un string gol.

La fel ca și Spring, React are un ecosistem vast. La acest ecosistem contribuie foarte mult comunitatea care aduce constant componente refolosibile care se pot utiliza foarte ușor. Cele mai utilizate librării din acest ecosistem care ajută la îmbunătățirea productivității sunt: “React Router” care se ocupă cu rutarea pentru URL-urile paginilor din React, prin această librările se poate securiza accesul către pagini cu ajutorul rolurilor de utilizator, “Redux” librărie folosită pentru administrarea stării aplicației și “Axios” pentru cererile HTTP.

2.7.1 Axios

Axios este o librările React folosită pentru a face cereri HTTP dintr-un browser web la un server. Oferă un mod simplu si intuitiv pentru programatori de a apela un API și a de a se ocupa cu operații asincrone în comunicarea cu un server.

Axios se bazează pe un model bazat pe Promises (promisiuni), ceea ce însemnă ca pentru fiecare cerere HTTP facută astfel se returnează un Promise. Un promise este un obiect care reprezintă modul în care o operație asincronă a decurs, în general aducerea informației de la un server, adică poate să rezulte intr-un răspuns de la server sau o eroare în momentul în care ceva nu a decurs bine. Mai departe un Promise pune la dispoziția programatorului un mod în care poate să gestioneze raspunsul primit și să fie salvat intr-o componentă sau un mod în care poate vedea eroarea primită împreună cu alte detalii legate de aceasta pentru a putea fi rezolvată cat mai ușor. Pentru cazul în care se primește răspunsul de la server se utilizează funcția “.then()” iar pentru cazul în care se primește o eroare există funcția “.catch()” . Aceste funcții se folosesc concomitent pentru a se acoperii ambele cazuri la fiecare cerere HTTP.

Pentru a putea folosii Axios este necesară instalarea, care se poate realiza printr-un manager de pachete cum ar fi: npm. După instalare se poate importa în paginile în care este nevoie de cererile HTTP. Acestă librărie are suport pentru toate browserele importante și funcționează după aceleași reguli în fiecare penru o consistență în răspunsuri.

Modul în care se fac cererile HTTP este intuitiv astfel că pentru metodele de bază CRUD (POST, GET, PUT, DELETE) sintaxa axios arată astfel: “axios.post()” , “axios.get()” și asa mai departe. Configurările cererilor se pot face prin opțiunile adiționale: headers, query parameters, request data și timeouts.

2.7.2 Redux

Redux este o librărie JavaScrit pentru managementul stării unei aplicații care se folosește în general cu framework-uri ca și React sau Angular. Cu această librărie se creează un container pentru starea aplicației în care se rețin diferite detalii cum ar fi datele utilizatorului curent pentru fiecare sesiune de pe fiecare dispozitiv unde funcționează aplicația.

Conține o analiză a ceea ce s-a realizat/studiat anterior. Arătați că ați studiat materiale bibliografice și că ați înțeles ceea ce ați citit.

Puteți include diferite puncte de vedere asupra problemei pe care o rezolvați în lucrare.

Nu uitați să citați corespunzător autorii oricărei idei extrase dintr-o sursă bibliografică.

# Analiză, proiectare, implementare

Aceasta parte a lucrării este flexibilă și depinde foarte mult de natura lucrării, poate fi organizată în mai multe capitole și conține contribuțiile personale ale autorului.

Includeți:

* + Detalii referitoare la analiză și proiectare:
    - descrierea metodelor pe care le-ați aplicat pentru rezolvarea problemei,
    - descrierea materialelor, procedurilor
    - calcule, tehnici, descrierea echipamentelor
    - metodologia de proiectare
    - informațiile necesare pentru ca cineva să poată reface lucrarea
  + Implementare :
    - Descrieți detaliile tehnice ale implementării aplicației: mediul de implementare, modul de prezentare, modul de utilizare al aplicației, etc.
  + Testare si validare :
    - Descrieți metodologia de testare a aplicației și rezultatele
    - Includeți experimentele pe care le-ați realizat, analiza rezultatelor pe care le-ați obținut.

# Concluzii

## Rezultate obținute

Evidențiați toate rezultatele pe care le-ați obținut și trageți concluzii din ele. Puteți prezenta o analiză critică a ceea ce ați realizat comparativ cu alte lucrări/studii anterioare.

Includeți o listă a contribuțiilor pe care le-ați avut în domeniul temei abordate.

## Direcții de dezvoltare

Descrieți direcțiile posibile de dezvoltare.

# Reguli de formatare

## Formatarea paginii

* + Dimensiunea paginii: A4
  + Margini: 2.5 cm (sus, jos, stânga, dreapta)
  + Antet și subsol: 1.27 cm de la marginea paginii
  + În antetul paginii (header): titlul capitolului, centrat, stil: Header\_style
  + În subsolul paginii: numărul paginii, centrat

## Titluri și stiluri

Titlurile capitolelor și subcapitolelor se marchează cu stilurile Heading 1 – 4, conform documentului model anexat în format Word. Descrierea stilurilor utilizate în document este prezentată în Tabelul 5.1.

Tabelul 5.1. Stiluri utilizate în acest document

| Nr. | Stil | Utilizat pentru | Format |
| --- | --- | --- | --- |
| 1 | Normal | Text normal | Font: (Default) Cambria, 12 pt, Justified, Line spacing: Multiple 1.1 li, Space After: 6 pt |
| 2 | Titlu | Titlul proiectului, prima pagină | Font: 24 pt, Small caps, Centered Line spacing: single, Space Before: 126pt, After: 0 pt, |
| 3 | Titlu2 | Titlul proiectului, pagina de prezentare | Font:14pt, Bold, Centered |
| 4 | Heading 1 | Titlurile capitolelor (nivel 1) | Font: 24 pt, Indent: Left: 0 cm Hanging: 0.76 cm, Space Before: 24pt, After: 12pt |
| 5 | Heading 2 | Titlurile subcapitolelor (nivel 2) | Font: 14 pt, Bold, Indent: Left: 0 cm  Hanging: 1.02 cm, Space Before: 18pt, After: 12pt |
| 6 | Heading 3 | Titlurile secțiunilor (nivel 3) | Font: Bold, Indent: Left: 0 cm Hanging: 1.27 cm, Space Before: 6 pt, After: 6pt |
| 7 | Heading 4 | Titlurile secțiunilor (nivel 4) | Font: Italic, Indent: Left: 0 cm Hanging: 1.52 cm, Space Before: 2 pt, After: 0 pt |
| 8 | Caption | Legenda figurilor și tabelelor | Font: Italic, Font color: Text 1, Line spacing: single, Space After: 10 pt, |
| 9 | Header\_style | Antetul paginii | Font: 10 pt, Italic, Centered, Border: Bottom: (Single solid line, Background 1, 0.5 pt Line width) |

## Figuri, tabele și ecuații

### Figuri

Figurile se inserează în text centrate, cu etichetă de numerotare și legendă (Caption) în partea de jos a figurii. Numărul figurii include și numărul capitolului, după exemplul prezentat în Figura 5.1.



Figura 5.1. Figură exemplu, stil: Caption

## Tabele

Tabelele se inserează în text centrate, cu etichetă și legendă (Caption) în partea de sus a tabelului, aliniată la stânga. Numărul tabelului include și numărul capitolului, după cum este prezentat, de exemplu, în Tabelul 5.1.

## Ecuații

Ecuațiile se inserează în text centrate, cu numerotare în partea dreaptă. Numărul ecuației include și numărul capitolului, conform exemplului din relația (5.1).

|  |  |
| --- | --- |
|  | (5.1) |

## Referințe bibliografice

Se recomandă ca citarea referințelor bibliografice să fie făcută în formatul IEEE.

În secțiunea Bibliografie sunt prezentate exemple pentru: o citare a unui capitol dintr-o carte [1], un articol publicat într-o revistă [2] și un articol publicat la o conferință [3].

Detalii cu privire la formatul citării diverselor tipuri de referințe pot fi găsite în [4] sau [5].

Referințele bibliografice se pot insera în text utilizând facilitățile Word de a adăuga surse și bibliografie unui document (References -> Citations & Bibliography). Dacă formatul IEEE pentru bibliografie nu este instalat implicit în Word, se poate descărca gratuit de la:

<https://bibword.codeplex.com/wikipage?title=Styles&referringTitle=Home>

Instrucțiunile de instalare pentru diferite versiuni de Word se pot obține de la aceeași adresă.

# Bibliografie

|  |  |
| --- | --- |
| [1] | P. Nume, "Titlul capitolului," în *Titlul cartii*, Oras, Editura, 2016, pp. 1-24. |
| [2] | P. Nume, "Titlul articolului," *Titlul revistei,* vol. 1, no. 2, pp. 22-30, 2016. |
| [3] | P. Nume, "Titlul articolului," în *Numele conferintei*, Oras, 2015. |
| [4] | "IEEE Citation Reference," 2009. [Online]. Available: https://www.ieee.org/documents/ieeecitationref.pdf. |
| [5] | "IEEE Editorial Style Manual," 2016. [Online]. Available: https://www.ieee.org/documents/style\_manual.pdf. |