



ADRIANA CĂLINOIU

ADRIANA GÎJU

ALEXANDRA MARCU

ELENA PREDEȘEL

MANAGEMENTUL LUCRĂRILOR DE
CERTIFICARE A COMPETENȚELOR
PROFESIONALE (ATESTAT) LA
PROFILUL MATEMATICĂ-
INFORMATICĂ

CRAIOVA
2025

ADRIANA CĂLINOIU
ADRIANA GÎJU

ALEXANDRA MARCU
ELENA PREDEȘEL

MANAGEMENTUL LUCRĂRILOR DE
CERTIFICARE A COMPETENȚELOR
PROFESIONALE (ATESTAT) LA
PROFILUL MATEMATICĂ-
INFORMATICĂ

CRAIOVA

2025

CUPRINS

1. Introducere	1
2. Alegerea temei lucrării de certificare a competențelor profesionale (atestat)	4
4. Documentarea și structura teoretică	13
5. Proiectarea aplicației	16
6. Planificarea structurii codului	19
6. Implementarea lucrării	22
7. Testarea și validarea proiectului	25
8. Redactarea lucrării finale	28
9. Prezentarea lucrării în fața comisiei	32
10. Rolul profesorului coordonator	36
11. Etică și originalitate	39
12. Exemple de teme pentru lucrări de atestat	42
13. Exemple de plan de lucru (model)	43
14. Erori frecvente și cum pot fi evitate	43
15. Checklist final pentru elevi	47
16. Checklist pentru profesorul coordonator	47
17. Anexe utile	47
18. Concluzii	49
19. BIBLIOGRAFIE	50

1. Introducere

Examenul de certificare a competențelor profesionale (atestat) se desfășoară conform Anexei la Ordinul MEN nr. 4433/ 29.08.2014 privind aprobarea Metodologiei de organizare și desfășurare a examenului de certificare a calificării absolvenților învățământului liceal, filiera vocațională, cu modificările și completările ulterioare.

Certificarea competențelor profesionale reprezintă un proces formal prin care o instituție autorizată verifică, evaluează și confirmă faptul că o persoană deține cunoștințele, abilitățile și aptitudinile necesare pentru a exercita o anumită ocupație sau activitate profesională, conform unor standarde naționale sau internaționale. Procesul presupune evaluarea atât a competențelor teoretice, cât și a celor practice, prin probe standardizate, portofolii, interviuri sau demonstrații practice.

Obținerea unui **certificat de competențe** oferă recunoaștere oficială a pregătirii unei persoane, indiferent dacă aceasta a fost dobândită în contexte formale, nonformale sau prin experiență la locul de muncă. Documentul este util pentru accesul la piața muncii, mobilitate profesională, continuarea studiilor sau avansare în carieră. Certificarea asigură angajatorilor că persoana evaluată îndeplinește cerințele ocupației și deține nivelul de profesionalism necesar.

Certificarea competențelor profesionale este important deoarece:

1. Pentru persoane:

- oferă recunoaștere oficială a cunoștințelor dobândite prin educație formală, informală sau experiență de muncă;
- crește șansele de angajare;
- contribuie la mobilitatea profesională în țară și în străinătate;
- permite accesul la salarii mai mari și la oportunități de avansare.

2. Pentru angajatori:

- garantează că angajatul are pregătirea necesară pentru ocupația respectivă;
- reduce costurile de formare internă;
- crește calitatea serviciilor și a producției.

3. Pentru sistemul educațional:

- aliniază programele de învățare cu nevoile pieței muncii;
- susține transparența și comparabilitatea calificărilor.

În urma susținerii examenului de certificare a competențelor, candidatul poate primi:

- certificat de competențe profesionale, dacă dovedește că îndeplinește toate cerințele;
- certificat de calificare profesională, în cazul calificărilor complete.

Aceste documente sunt recunoscute la nivel național și, în multe situații, la nivel european, facilitând angajarea sau continuarea studiilor.

Conform Metodologiei de organizare și desfășurare a examenului de certificare a calificării absolvenților învățământului liceal, filiera vocațională, cu modificările și completările ulterioare:

”Art. 22. – (1) Examenul de certificare conține două probe:

- a) susținerea proiectului/lucrării de specialitate;
- b) proba practică, relevantă pentru specializarea certificată.

(2) Proba practică poate avea mai multe etape.

(3) Probele examenului de certificare prevăzute la alin.(1), se pot desfășura în aceeași zi.”

” Art. 24. – Proiectul/lucrarea de specialitate reflectă cunoștințele și competențele generale și specifice dobândite de către elev, atât din punct de vedere teoretic cât și practic și își propune să verifice capacitatea candidatului de:

- a) a sistematiza și a sintetiza cunoștințele generale în domeniu și pe cele de specialitate corespunzătoare specializării absolvite;
- b) a utiliza deprinderile și competențele formate în rezolvarea problemelor teoretice și practice, specifice specializării absolvite;
- c) a mobiliza competențele specifice dobândite prin studierea disciplinelor din aria curriculară de specialitate în vederea rezolvării unor cazuri/situații problematice impuse de tema lucrării, după caz;
- d) a utiliza, după caz, tehnologiile informatice moderne în aplicații specifice domeniului în care s-a specializat.

Art. 25. – (1) Temele proiectelor/lucrărilor de specialitate și criteriile de evaluare sunt stabilite la nivelul unității de învățământ, de comisia metodică de specialitate care, acolo unde este necesar, poate consulta reprezentanți ai partenerilor sociali interesați.

(2) Temele proiectelor/lucrărilor de specialitate trebuie să se încadreze în mod obligatoriu în domeniul profilului respectiv și să fie adecvate programelor școlare studiate de elevi la disciplinele din ariile curriculare specifice specializării.

(3) Tematica probelor examenului de certificare se afișează la avizierul unității de învățământ până la data de 1 noiembrie a anului școlar în care se susține examenul.

(4) Candidații pot opta și pentru teme care nu se regăsesc în listă, dacă au acordul profesorului coordonator al proiectului/lucrării de specialitate și al responsabilului catedrei de specialitate. Opțiunea se face în scris, până la data de 10 noiembrie a anului școlar în care se susține examenul.

(5) Consiliul de administrație al unității de învățământ aprobă, până la data de 15 noiembrie a anului școlar în care se susține examenul repartizarea temelor/lucrărilor proiectelor de specialitate.”

Lucrarea de certificare a competențelor profesionale (atestat) la profilul Matematică-Informatică reprezintă una dintre cele mai importante etape din parcursul educațional al elevilor. Ea oferă oportunitatea de a demonstra competențe acumulate în timpul liceului, precum gândirea algoritmică, programarea, proiectarea sistemelor informatice și capacitatea de documentare. Un management eficient al lucrării de atestat asigură o experiență organizată, coerentă și orientată spre rezultate solide.

Acest ghid oferă elevilor și profesorilor coordonatori un cadru clar privind etapele, responsabilitățile, structura și evaluarea lucrării, astfel încât întregul proces să fie unul bine gestionat și de succes.

2. Alegerea temei lucrării de certificare a competențelor profesionale (atestat)

Alegerea temei lucrării de certificare a competențelor profesionale (atestat) reprezintă primul pas esențial în realizarea proiectului final. Tema trebuie selectată astfel încât să reflecte interesele elevului, competențele dobândite pe parcursul studiilor și specificul profilului „Matematică-Informatică”. O temă bine aleasă permite demonstrarea eficientă a cunoștințelor teoretice, a abilităților practice de programare sau analiză și a capacității de a rezolva probleme reale. De asemenea, tema trebuie să fie fezabilă, clar delimitată și relevantă atât din punct de vedere educațional, cât și profesional.

Conform metodologiei în vigoare

”Art. 24. – Proiectul/lucrarea de specialitate reflectă cunoștințele și competențele generale și specifice dobândite de către elev, atât din punct de vedere teoretic cât și practic și își propune să verifice capacitatea candidatului de:

a) a sistematiza și a sintetiza cunoștințele generale în domeniu și pe cele de specialitate corespunzătoare specializării absolvite;

b) a utiliza deprinderile și competențele formate în rezolvarea problemelor teoretice și practice, specifice specializării absolvite;

c) a mobiliza competențele specifice dobândite prin studierea disciplinelor din aria curriculară de specialitate în vederea rezolvării unor cazuri/situații problematice impuse de tema lucrării, după caz;

d) a utiliza, după caz, tehnologiile informatice moderne în aplicații specifice domeniului în care s-a specializat.

Art. 25. – (1) Temele proiectelor/lucrărilor de specialitate și criteriile de evaluare sunt stabilite la nivelul unității de învățământ, de comisia metodică de specialitate care, acolo unde este necesar, poate consulta reprezentanți ai partenerilor sociali interesați.

(2) Temele proiectelor/lucrărilor de specialitate trebuie să se încadreze în mod obligatoriu în domeniul profilului respectiv și să fie adecvate programelor școlare studiate de elevi la disciplinele din ariile curriculare specifice specializării.

(3) Tematica probelor examenului de certificare se afișează la avizierul unității de învățământ până la data de 1 noiembrie a anului școlar în care se susține examenul.

(4) Candiđații pot opta și pentru teme care nu se regăsesc în listă, dacă au acordul profesorului coordonator al proiectului/lucrării de specialitate și al responsabilului catedrei de

specialitate. Opțiunea se face în scris, până la data de 10 noiembrie a anului școlar în care se susține examenul.

(5) Consiliul de administrație al unității de învățământ aprobă, până la data de 15 noiembrie a anului școlar în care se susține examenul repartizarea temelor/lucrărilor proiectelor de specialitate.”

Tema trebuie deci să permită evidențierea competențelor dobândite în anii de studiu: gândire algoritmică, programare, modelare matematică, utilizarea bazelor de date, proiectare web sau analiză informatică.

Alegerea temei este primul pas esențial. Tema trebuie să fie:

- **Relevantă** pentru domeniul informatic demonstrând competențele prevăzute în standardele de formare;
- **Adecvată nivelului de pregătire** al elevului;
- **Clară și bine delimitată**, evitând subiectele prea generale sau prea complexe;
- **Fezabilă**, astfel încât să poată fi finalizată în timpul disponibil deci realizabilă într-un interval limitat de timp și cu resurse accesibile;
- **Stimulativă**, pentru a menține motivația elevului;
- **Utilă**, ideal cu o componentă aplicată sau practică.

Surse posibile de inspirație:

- Aplicațiile învățate prin participarea la proiectul nr. **2024 - 1 - RO01 - KA122 - VET 000217671 Transformarea Educației Militare prin Competente Informactice Avansate**;
- Aplicații software educaționale;
- Jocuri simple și medii de simulare;
- Aplicații mobile;
- Baze de date pentru evidențe (elevi, produse, activități);
- Analiză matematică sau modelare numerică;
- Mini-platforme web sau aplicații pentru automatizare.

Profesorul coordonator aprobă tema și oferă orientările inițiale privind complexitatea și direcția de lucru.

Profesorul coordonator are un rol important în ghidarea elevului spre o temă potrivită, oferind feedback privind nivelul de dificultate, relevanța și utilitatea acesteia. O temă bine aleasă

facilitează planificarea etapelor lucrării, structurarea corectă a proiectului și realizarea unei implementări tehnice coerente.

În final, alegerea temei reprezintă o oportunitate pentru elev de a-și valorifica pasiunile, de a demonstra maturitatea profesională și de a crea un proiect care să îi reflecte potențialul în perspectiva continuării studiilor sau integrării pe piața muncii.

3. Stabilirea unui plan de lucru

Stabilirea unui plan de lucru reprezintă etapa care asigură coerența, organizarea și realizarea eficientă a lucrării de atestat. După alegerea temei, elevul, împreună cu profesorul coordonator, elaborează un calendar clar al activităților, care permite structurarea procesului de lucru, respectarea termenelor și evitarea suprasolicitării în perioada finală.

Un plan de lucru bine realizat începe cu **definirea obiectivelor principale** ale proiectului: ce se dorește obținut, ce funcționalități trebuie implementate, ce cunoștințe trebuie demonstrate. Aceste obiective sunt apoi împărțite în sarcini mai mici și mai ușor de gestionat. De exemplu, pentru o aplicație software: studiu teoretic, proiectarea interfeței, definirea claselor, scrierea codului, testare și documentare.

Un alt element esențial este **stabilirea unui calendar realist**, adaptat ritmului elevului și dificultății proiectului. Calendarul include etape precum: documentarea inițială, redactarea capitolelor teoretice, realizarea părții practice, corecturi, finalizare și pregătirea prezentării. Fiecare etapă are un termen clar și un set de livrabile (de exemplu: un capitol finalizat, o funcție testată, un model de date realizat).

Planul de lucru trebuie să conțină și **resursele necesare**, cum ar fi software-ul folosit, bibliotecile, datele de test, manualele sau tutorialele tehnice. Acestea se stabilesc de la început pentru a evita întârzierile cauzate de căutarea ulterioară a materialelor.

Profesorul coordonator are rolul de a valida planul și de a se asigura că acesta este echilibrat și realizabil. Pe parcursul lucrului, planul poate fi ajustat în funcție de dificultățile întâmpinate sau de evoluția proiectului, păstrând însă structura de bază.

Un plan de lucru bine conceput devine astfel **o hartă a întregului proces**, oferind claritate, ritm și disciplină. El ajută elevul să lucreze organizat, să își gestioneze timpul eficient și să finalizeze proiectul cu un nivel înalt de calitate, transformând etapa de realizare a lucrării într-un proces predictibil și profesional.

Planul de lucru include toate elementele necesare pentru organizarea și finalizarea în condiții optime a lucrării de atestat. El funcționează ca un document de orientare, menit să ofere claritate, ordine și coerență pe parcursul întregului proces. Un plan bine structurat oferă elevului o imagine de ansamblu asupra sarcinilor ce trebuie îndeplinite și ajută la respectarea termenelor, evitând întârzierile și suprasolicitarea.

În primul rând, planul de lucru include **definirea obiectivelor proiectului**. Acestea stabilesc direcția generală a lucrării: ce competențe trebuie demonstrate, ce rezultat final se urmărește și ce relevanță are proiectul în raport cu tema aleasă. Obiectivele sunt formulate clar și specific, astfel încât elevul să știe exact ce trebuie să atingă.

În al doilea rând, planul cuprinde **etapele principale ale realizării lucrării**, împărțite în activități specifice. De exemplu:

- documentarea teoretică;
- crearea structurii capitolelor;
- proiectarea și implementarea aplicației;
- scrierea codului și testarea funcționalităților;
- redactarea părții practice și realizarea anexelor;
- corectarea și formatarea finală.

Aceste etape sunt prezentate în ordine logică, reflectând procesul real de dezvoltare a proiectului.

Planul de lucru include și **un calendar detaliat**, în care sunt precizate termenele-limită pentru fiecare activitate. Acest calendar ajută elevul să își distribuie volumul de muncă într-un mod echilibrat, să respecte ritmul de lucru și să finalizeze proiectul în timp util. Calendarul este flexibil și poate fi ajustat dacă apar dificultăți sau întârzieri, însă menține direcția principală a procesului.

Un alt element important îl reprezintă **resursele necesare**: programe software, limbaje de programare, baze de date, biblioteci, documentație tehnică, echipamente sau acces la materiale suplimentare. Stabilirea acestor resurse înainte de începerea lucrului permite o desfășurare fluentă a activităților.

Planul de lucru poate include și **posibile riscuri sau dificultăți**, precum probleme tehnice, lipsa datelor de test sau nevoia de aprofundare a unor concepte. Identificarea lor de la început ajută la o mai bună pregătire și prevenire.

În final, planul cuprinde **forma de monitorizare și comunicare cu profesorul coordonator**, prin întâlniri periodice, prezentarea progresului sau trimiterea de versiuni intermediare ale lucrării.

Astfel, planul de lucru include toate elementele necesare pentru ca elevul să lucreze organizat, să își gestioneze timpul eficient și să finalizeze lucrarea de atestat într-un mod profesionist și coerent.

Deci, în concluzie, un plan de lucru clar ajută elevul să își gestioneze timpul și resursele eficient. Acesta include:

- Calendarul etapelor principale;
- Obiective intermediare (documentare, proiectare, implementare);
- Termene limită;
- Sesiuni de consultare cu profesorul coordonator.

Recomandări de organizare:

Reprezintă un set de bune practici care îi ajută pe elevi să își gestioneze eficient timpul, resursele și activitățile pe parcursul realizării lucrării de atestat. Aceste recomandări contribuie la creșterea calității proiectului, la reducerea stresului și la organizarea unui proces de lucru coerent și profesionist.

În primul rând, se recomandă **planificarea riguroasă a etapelor**. Elevul ar trebui să stabilească un program clar, împărțind proiectul în segmente mici, ușor de gestionat, și să respecte calendarul stabilit. Planificarea include momente de documentare, redactare, programare și verificare, astfel încât volumul de lucru să fie distribuit uniform pe întreaga perioadă.

A doua recomandare esențială este **menținerea unei documentări constante**. Elevii ar trebui să consulte surse actuale și relevante, să își creeze un fișier special cu linkuri, citate și resurse utile, și să noteze orice idee importantă care poate contribui la dezvoltarea proiectului. O documentare organizată economisește timp și crește calitatea argumentelor teoretice.

Un aspect important îl constituie și **organizarea eficientă a fișierelor**. Toate materialele — cod sursă, imagini, grafice, explicații, versiuni intermediare — ar trebui păstrate în foldere clar denumite. Este utilă și salvarea periodică în cloud sau pe alte dispozitive pentru a evita pierderea datelor.

O altă recomandare este **realizarea și testarea progresivă a părții practice**. Elevii ar trebui să implementeze funcționalitățile treptat, verificând fiecare componentă înainte de a trece la următoarea. Această abordare reduce riscul erorilor acumulate și permite corectarea rapidă a problemelor.

În plus, este esențială **comunicarea constantă cu profesorul coordonator**. Întâlnirile periodice, prezentarea versiunilor intermediare și solicitarea de feedback ajută la menținerea direcției corecte a proiectului și la clarificarea eventualelor neînțelegeri sau dificultăți.

Se recomandă, de asemenea, alocarea unui timp special pentru **revizuirea finală**. Aceasta include corectarea greșelilor de formatare, stil, ortografie, verificarea funcționalității aplicației și pregătirea anexelor. O lucrare bine revizuită reflectă profesionalism și responsabilitate.

Nu în ultimul rând, elevilor li se recomandă să adopte **o disciplină de lucru constantă**, evitând amânările. Realizarea câtorva sarcini mici în fiecare zi este mult mai eficientă decât lucrul intensiv pe ultima sută de metri.

Astfel, recomandările de organizare îi ajută pe elevi să își structureze eforturile, să lucreze ordonat și eficient și să ajungă la un rezultat final complet, coerent și de calitate.

Recomandări de organizare – Listă clară și practică

1. Planificarea timpului

- Împarte proiectul în etape mici și realizabile.
- Stabilește termene realiste pentru fiecare etapă.
- Evită amânarea — lucrează câte puțin în mod constant.
- Folosește un calendar digital sau o agendă pentru monitorizarea progresului.

2. Organizarea documentării

- Creează un folder cu toate sursele teoretice.
- Notează ideile importante și bibliografia pe măsură ce le găsești.
- Folosește doar surse credibile: manuale, articole academice, site-uri specializate.

3. Structurarea eficientă a fișierelor

- Organizează fișierele în foldere distincte: *Teorie, Proiect, Cod, Resurse, Anexe*.
- Denumeste clar fiecare document.
- Salvează periodic o copie de siguranță în cloud (Google Drive, OneDrive etc.).

4. Dezvoltarea progresivă a părții practice

- Începe cu funcționalitățile principale, apoi adaugă elementele secundare.
- Folosește rezultatele proiectului VET - Proiect nr. **2024 - 1 - RO01 - KA122 - VET 000217671 Transformarea Educației Militare prin Competente Informatică Avansate** care a creat șanse reale elevilor având specializarea Matematică-Informatică de a dobândi competențe profesionale și transversale necesare pentru a îmbrățișa noile tehnologii în cariera militară. Prin inovarea și internaționalizarea formării se va construi o educație activ-participativă, practică și adaptată cerințelor sistemelor militare actuale pentru realizarea obiectivului strategic național **„România sigură și**

prosperă într-o lume marcată de provocări”.

- Testează pe măsură ce implementezi.
- Păstrează o versiune stabilă a proiectului la fiecare etapă importantă.

5. Colaborarea cu profesorul coordonator

- Programează discuții regulate.
- Prezintă variante intermediare ale proiectului.
- Cere feedback pentru claritate, structură și acuratețe tehnică.

6. Revizuirea finală

- Corectează greșelile de formatare și ortografie.
- Verifică toate referințele și anexele.
- Testează încă o dată aplicația sau modelul realizat.
- Pregătește prezentarea orală și materialele vizuale.

Checklist de organizare pentru lucrarea de atestat

I. Alegerea și definirea proiectului

- Am ales o temă clară și relevantă pentru profilul meu.
- Am discutat și aprobat tema cu profesorul coordonator.
- Am formulat obiectivele principale ale lucrării.

II. Planificarea

- Am împărțit proiectul în etape logice.
- Am creat un calendar al activităților.
- Am stabilit resursele necesare (software, date, bibliografie etc.).

III. Documentarea

- Am colectat surse teoretice relevante.
- Am început redactarea notițelor și bibliografiei.
- Am salvat sursele în foldere organizate.

IV. Dezvoltarea proiectului practic

- Am aplicat rezultatele proiectului **Transformarea Educației Militare prin Competente Informactice Avansate pentru** dobândirea competențelor informatice în vederea realizării unui atestat profesional complex, adaptat noilor tehnologii și inserării adecvate dezvoltării profesionale, în școlile militare și în Armata Română.
- Am proiectat structura aplicației/modelului.

- Am implementat funcționalitățile principale.
- Am testat fiecare etapă.
- Am ajustat codul/structura în urma testării.

V. Redactarea lucrării

- Am scris partea teoretică.
- Am redactat partea practică.
- Am creat anexele (capturi, diagrame, cod relevant etc.).
- Am respectat normele de formatare (font, margini, numerotarea paginilor).

VI. Revizuirea

- Am verificat întregul document pentru greșeli.
- Am primit feedback final de la profesor.
- Am pregătit versiunea finală pentru predare.

VII. Pregătirea prezentării

- Am realizat prezentarea (PowerPoint/Canva).
- Am repetat expunerea orală.
- Am pregătit demonstrația practică.

4. Documentarea și structura teoretică

Reprezintă fundamentul conceptual al lucrării de atestat, oferind cadrul necesar pentru înțelegerea, argumentarea și susținerea părții practice. Această etapă presupune atât procesul de colectare și analiză a informațiilor relevante, cât și organizarea lor într-o formă coerentă, academică și adaptată temei proiectului.

În primul rând, documentarea constă în **identificarea, selectarea și studierea surselor de informare**: manuale, cărți de specialitate, articole științifice, resurse online verificate, tutoriale, documentații oficiale ale limbajelor de programare sau ale bibliotecilor software utilizate. Elevul trebuie să caute informații actualizate, cu un nivel adecvat de credibilitate și relevanță. Pe parcurs, este necesar să noteze ideile importante, definițiile, conceptele-cheie și relațiile dintre acestea pentru a putea construi un material teoretic solid.

Un alt aspect important este **organizarea și citarea corectă a surselor**. Toate informațiile preluate trebuie integrate într-un mod personal, argumentat, evitând copierea directă. Citarea bibliografică — inclusă la finalul lucrării — conferă seriozitate, rigoare academică și respect față de principiile etice ale cercetării.

În ceea ce privește structura teoretică, aceasta trebuie să fie **clară, logică și adaptată temei**. În mod obișnuit, partea teoretică include următoarele elemente:

- **Introducere în domeniu**: prezentarea generală a temei, motivația alegerii și contextul în care se înscrie lucrarea.
- **Concepte de bază**: explicații ale noțiunilor fundamentale (ex.: algoritmi, structuri de date, modele matematice, limbaje de programare), necesare pentru înțelegerea ulterioară a proiectului.
- **Prezentarea tehnologiilor și instrumentelor utilizate**: descrierea resurselor software, a paradigmelor de programare sau a metodelor matematice relevante.
- **Fundamente teoretice ale aplicației**: modul în care teoria susține partea practică, justificând alegerea soluțiilor tehnice.

Structura teoretică trebuie să fie redactată într-un stil clar, concis și bine argumentat, evitând excesul de definiții fără relevanță pentru proiect. Scopul acesteia este să demonstreze că elevul înțelege temelia conceptuală a lucrării și poate explica deciziile tehnice bazându-se pe principii teoretice.

În final, documentarea și partea teoretică oferă lucrării de atestat **coerență, credibilitate și profunzime**, constituind suportul necesar pentru implementarea practică și pentru prezentarea finală.

Orice lucrare de atestat trebuie să conțină o componentă teoretică ce demonstrează înțelegerea conceptelor utilizate. Elevul va consulta:

- Manuale de specialitate;
- Cursuri online;
- Documentația limbajelor și tehnologiilor folosite;
- Articole și resurse academice.

Structura recomandată a componentei teoretice:

1. Introducere în tema proiectului;
2. Fundamentele teoretice ale domeniului - prezentarea tehnologiilor folosite (ex.: C++, Python, SQL, HTML/CSS/JS);
3. Tehnologii utilizate – prezentare aplicație;
4. Concluzii parțiale asupra designului soluției.

MODEL DE STRUCTURĂ TEORETICĂ

CAPITOLUL I. Introducere în tema proiectului

1.1. Contextul general al temei

- Prezentarea domeniului (ex.: programare, baze de date, aplicații web, modelare matematică).
- Situația actuală a utilizării tehnologiei în domeniul abordat.
- Importanța temei în educație, industrie sau viața de zi cu zi.

1.2. Motivația alegerii temei

- De ce este tema relevantă pentru elev.
- Utilitatea practică a proiectului.
- Legătura cu competențele dobândite în liceu (algoritmică, OOP, web design, modelare etc.).

1.3. Obiectivele lucrării

- Ce urmărește lucrarea să demonstreze.

- Ce abilități sunt evidențiate.
- Ce problemă practică sau teoretică se rezolvă.

CAPITOLUL II. Fundamente teoretice ale domeniului

2.1. Noțiuni generale

Aici se introduc conceptele esențiale pentru tema aleasă. Se folosesc și cunoștințele dobândite prin participarea la proiectul **Transformarea Educației Militare prin Competente Informactice Avansate**.

Exemple:

Pentru programare:

- algoritmi;
- structuri de date;
- paradigme de programare.

Pentru web:

- arhitectura client–server;
- HTML, CSS, JavaScript;
- noțiuni despre securitate.

Pentru baze de date:

- modele de baze de date;
- tipuri de relații;
- normalizare.

2.2. Terminologie specifică

Definirea termenilor-cheie utilizați în lucrare, cu explicații clare și exemple.

CAPITOLUL III. Tehnologii și instrumente utilizate

3.1. Limbajul de programare

- Istoric scurt.
- Caracteristici.
- Avantaje pentru proiectul ales.
- Exemple de utilizare.

3.2. Mediul de dezvoltare

- IDE sau platformă folosită (ex.: Python IDLE, Visual Studio Code, NetBeans, IntelliJ, XAMPP).
- Justificarea alegerii acestuia.

CAPITOLUL IV. Fundamente teoretice ale aplicației – Structura aplicației

4.1. Analiza problemei

- Descrierea problemei reale pe care o rezolvă aplicația.
- Identificarea cerințelor funcționale și nefuncționale.

4.2. Modelarea conceptuală

- Diagrame logice sau conceptuale (UML, ERD, diagrame de flux).
- Structuri matematice sau algoritmice necesare.

4.3. Soluția tehnică propusă

- Prezentarea arhitecturii (modul, componente, funcționalități).
- Justificarea alegerii soluției.

CAPITOLUL V. Concluzii

- Retrospectivă asupra noțiunilor prezentate.
- Legătura dintre teoria discutată și partea practică.
- Importanța cunoașterii bazei teoretice în realizarea proiectului.

5. Proiectarea aplicației

Proiectarea aplicației reprezintă etapa centrală în realizarea lucrării de atestat, în care ideile teoretice sunt transformate într-un plan concret pentru implementarea soluției software. Această fază este esențială deoarece organizează structura aplicației, definește funcționalitățile, stabilește tehnologiile utilizate și clarifică modul în care componentele sistemului vor interacționa. Un proiect bine realizat asigură eficiență, claritate și ușurință în programare, reducând riscul erorilor și al blocajelor ulterioare.

Aplicația va fi realizată și pe baza competențelor dobândite prin participarea la proiectul **Transformarea Educației Militare prin Competențe Informatică Avansate** - prin care elevii au fost instruiți în dezvoltarea de jocuri video, o competență esențială având în vedere că orice aplicație militară modernă este testată inițial prin simulări și jocuri video – de la îndeplinirea misiunilor militare, simulatoare de zbor, până la ședințe de tragere.

1. Analiza cerințelor aplicației

Primul pas în proiectare este identificarea și definirea clară a cerințelor:

- **Cerințe funcționale** – ce trebuie să facă aplicația (de ex. salvarea datelor, generarea rapoartelor, autentificare, căutare).
- **Cerințe nefuncționale** – performanță, securitate, accesibilitate, scalabilitate, ergonomie.

Aceste cerințe sunt formulate în acord cu tema aleasă și discutate cu profesorul coordonator.

Ele devin baza pentru toate deciziile tehnice viitoare.

2. Stabilirea arhitecturii aplicației

Arhitectura descrie structura generală a proiectului și modul în care componentele cooperează.

Aceasta poate include:

- aplicație *desktop* bazată pe module și clase;
- aplicație *web* cu separarea client–server;
- aplicație cu baze de date integrate;
- aplicație orientată pe obiecte, cu clase specificate clar.

Arhitectura determină modul în care datele circulă în aplicație și cum sunt realizate funcțiile principale.

3. Modelarea logică – diagrame și reprezentări

Un element important în proiectarea aplicației este folosirea unor instrumente de modelare, precum:

- **Diagrame de flux** – arată logica proceselor.
- **Diagrame UML**
 - diagrama de cazuri de utilizare (use case diagram);
 - diagrama de clase;
 - diagrama de secvență.
- **Scheme logice și algoritmi** – pentru procese specifice, cum ar fi sortarea, filtrarea sau generarea de rapoarte.
- **Modelarea bazei de date** – prin diagrame ERD (Entity–Relationship Diagram).

Aceste diagrame clarifică modul de funcționare a aplicației înainte ca programarea să înceapă.

4. Proiectarea interfeței utilizator

Interfața este componenta cu care utilizatorul interacționează direct. Această etapă implică:

- definirea ecranelor interfeței;
- stabilirea elementelor grafice (butoni, meniuri, câmpuri);
- stabilirea modului în care utilizatorul navighează în aplicație;
- mock-up-uri sau schițe preliminare.

O interfață intuitivă crește valoarea aplicației și facilitează testarea.

5. Alegerea tehnologiilor și a resurselor software

Pe baza analizei și modelării, se selectează:

- limbajul de programare (Python, C++, Java, PHP etc.);
- framework-uri sau biblioteci (Tkinter, Pygame, Bootstrap, MySQL, SQLite);
- software-ul necesar (IDE, server, editor grafic).

Alegerea tehnologiilor trebuie justificată teoretic în lucrare, explicând de ce sunt potrivite pentru funcționalitățile proiectului.

6. Planificarea structurii codului

În această etapă se definesc:

- clasele și obiectele principale;
- funcțiile și responsabilitățile fiecărei componente;
- modul în care componentele comunică între ele;
- regulile de organizare a proiectului (nomenclatură, directoare, fișiere).

Această clarificare previne suprapunerea funcțiilor și creează un cod ușor de extins și întreținut.

Planificarea structurii codului reprezintă o etapă esențială în realizarea aplicației, deoarece definește modul în care programul va fi organizat, clarifică responsabilitățile fiecărei componente și facilitează implementarea, testarea și întreținerea codului. O structură bine planificată reduce riscul erorilor, permite dezvoltarea modulară și face proiectul mai ușor de înțeles pentru coordonator și evaluatori.

1. Definirea componentelor principale

Primul pas este identificarea **componentelor sau modulelor** aplicației. Acestea pot fi:

- **Module funcționale:** grupuri de funcții care realizează sarcini similare (ex.: gestionarea utilizatorilor, calculul rezultatelor, generarea rapoartelor).
- **Clase și obiecte** (pentru programele orientate pe obiecte): fiecare clasă reprezintă o entitate relevantă (ex.: Elev, Profesor, Produs).
- **Fișiere și directoare:** organizarea fișierelor sursă, resurselor, bibliotecilor și documentației într-un mod logic.

Planificarea componentelor permite elevului să delimiteze clar responsabilitățile fiecărei părți din cod și să evite suprapunerea funcțiilor.

2. Definirea funcțiilor și responsabilităților

Pentru fiecare modul sau clasă se stabilesc:

- **Funcțiile principale:** ce sarcini va îndeplini componenta (ex.: adăugare, ștergere, modificare, afișare).
- **Interacțiunea între funcții:** care funcții apelează alte funcții sau module.
- **Intrările și ieșirile:** ce date primește și ce rezultate produce fiecare funcție.

Această planificare ajută la construirea unui cod modular, ușor de testat și întreținut.

3. Organizarea fișierelor și directoarelor

Structura codului trebuie să fie clară și coerentă. De exemplu:

- **src/** – fișiere sursă principale
- **lib/** – biblioteci sau funcții auxiliare
- **data/** – fișiere cu date de test sau baze de date
- **docs/** – documentație și diagrame
- **tests/** – fișiere pentru teste unitare

O structură organizată reduce confuzia, facilitează găsirea rapidă a codului și permite coordonatorului să înțeleagă mai ușor implementarea.

4. Convenții de numire și stil

Stabilirea de la început a unor reguli de denumire a variabilelor, funcțiilor, claselor și fișierelor este esențială:

- Nume descriptive pentru funcții și variabile (ex.: `adaugaElev()`, `calculeazaMedie()`)
- Folosirea consistentă a camelCase sau snake_case
- Comentarii clare pentru fiecare funcție și secțiune de cod

Aceasta facilitează citirea și înțelegerea codului, atât pentru elev, cât și pentru profesorul coordonator.

5. Gestionarea dependențelor și interacțiunilor

Planificarea codului include identificarea **interdependențelor între module** și modul în care acestea comunică. Exemple:

- Un modul de date poate fi apelat de modulul de interfață pentru afișarea informațiilor.
- Modulul de calcul poate primi date de la modulul de input și trimite rezultate către modulul de output.

Stabilirea acestor legături previne erorile logice și facilitează testarea.

6. Versiuni și control al codului

Elevii sunt încurajați să utilizeze:

- **Versiuni intermediare ale codului:** salvarea periodică a stadiilor de lucru.
- **Sisteme de versionare** (opțional, Git) pentru a urmări modificările și a putea reveni la o variantă anterioară.

Aceasta ajută la gestionarea proiectului, prevenind pierderea datelor și permițând corecturi rapide.

7. Sinteză

Planificarea structurii codului oferă:

- Claritate în implementare

- Organizare modulară
- Ușurință în testare și depanare
- Facilitează evaluarea lucrării de atestat

Această etapă asigură o bază solidă pentru implementarea practică și contribuie semnificativ la succesul final al proiectului.

7. Definirea scenariilor de utilizare

Pentru a verifica dacă aplicația este completă, se construiesc scenarii tipice:

- „Utilizatorul adaugă o înregistrare”;
- „Utilizatorul caută un elev după nume”;
- „Sistemul generează un raport”;
- „Datele sunt salvate în baza de date”.

Aceste scenarii devin ulterior baza pentru testarea aplicației.

8. Sinteza proiectării

Proiectarea aplicației se finalizează cu o prezentare a structurii stabilite:

- arhitectura completă;
- diagramele relevante;
- interfața planificată;
- fluxurile de date;
- lista funcționalităților validate.

Această sinteză devine ghidul dezvoltării și garantează că implementarea va fi coerentă și eficientă.

Înainte de implementare, elevul trebuie să realizeze:

- **Diagrama de arhitectură** a aplicației;
- **Structura bazei de date**, dacă este cazul;
- **Diagrame de flux** pentru algoritmi;
- **Interfața grafică (mock-up)** pentru aplicațiile cu UI.

Profesorul coordonator validează designul tehnic înainte de a trece la implementare.

6. Implementarea lucrării

Aceasta este partea centrală a atestatului. Ea reprezintă etapa în care proiectul trece din faza de planificare și proiectare în faza de realizare concretă. Aceasta este componenta centrală a lucrării de atestat, întrucât aici elevul demonstrează aplicarea cunoștințelor dobândite în domeniul informaticii, capacitatea de a utiliza algoritmi, structuri de date, limbaje de programare și instrumente software pentru a crea o aplicație funcțională. Implementarea trebuie să urmeze planul stabilit anterior, dar poate include ajustări sau optimizări apărute pe parcurs.

Recomandări:

- Codul trebuie să fie clar, documentat și structurat;
- Se vor folosi standarde bune de programare;
- Testarea programului trebuie făcută sistematic;
- Se păstrează o arhivă a tuturor versiunilor de lucru.

Limbaje și tehnologii uzuale:

- C/C++ pentru aplicații de consolă sau cu interfață simplă;
- Python pentru simulări și aplicații educaționale;
- Java/JavaFX pentru interfețe grafice;
- HTML/CSS/JS + PHP/MySQL pentru aplicații web.

1. Transpunerea proiectării în cod

În această etapă sunt create efectiv fișierele sursă, clasele, funcțiile și modulele planificate.

Elevul:

- creează structura directorului proiectului;
- implementează clasele și funcțiile conform diagramelor UML sau descrierilor din etapa de proiectare;
- stabilește conexiunile între module astfel încât logica aplicației să fie coerentă;
- integrează biblioteci externe (dacă este cazul), asigurând compatibilitatea lor.

Implementarea trebuie realizată gradual, pornind de la funcțiile de bază și continuând cu cele mai complexe.

2. Construirea interfeței aplicației

Dacă lucrarea include o interfață grafică (GUI sau web), elevul:

- creează ecranele principale (login, meniu, panou administrativ etc.);
- implementează interacțiunile utilizatorului cu aplicația;

- conectează interfața cu logica programului;
- se asigură că designul este intuitiv și utilizabil.

Pentru aplicațiile consolă, se definesc meniuri text clare și structurate.

3. Implementarea funcțiilor principale

Elevul implementează funcționalitățile de bază stabilite inițial, precum:

- adăugarea, modificarea și ștergerea datelor;
- operații de calcul (statistici, rapoarte, sortări, filtrări);
- generarea de fișiere (PDF, CSV etc., dacă este cazul);
- procesarea datelor introduse de utilizator.

Fiecare funcționalitate trebuie testată imediat după implementare pentru a preveni erorile cumulative.

4. Gestionarea datelor

Dacă aplicația folosește o bază de date sau fișiere:

- se creează structura bazei de date (tabele, relații, câmpuri);
- se implementează conexiunea cu programul;
- se creează funcții de inserare, actualizare, interogare și ștergere;
- se implementează validări pentru a preveni introducerea de date eronate.

Pentru fișiere locale, se stabilesc formatele de stocare (TXT, JSON, CSV).

5. Tratarea erorilor și validarea datelor

Un proiect bine implementat include:

- verificări la introducerea datelor;
- mesaje de eroare clare și utile;
- gestionarea situațiilor speciale (ex.: câmpuri goale, date ilogice, fișiere lipsă);
- asigurarea stabilității aplicației în condiții neprevăzute.

6. Integrarea și testarea incrementală

Implementarea nu se face într-un singur bloc, ci progresiv:

1. codarea unei funcționalități;
2. testarea ei imediată;
3. trecerea la următoarea funcție.

Această abordare reduce riscul erorilor majore și ajută la înțelegerea comportamentului aplicației.

7. Optimizări și îmbunătățiri

În timpul implementării, elevul poate identifica soluții mai bune decât cele inițial planificate.

Optimizările pot include:

- reducerea timpului de execuție al anumitor funcții;
- simplificarea codului;
- reorganizarea logicii interne a unor module;
- îmbunătățirea experienței utilizatorului.

Aceste ajustări trebuie menționate și motivate în lucrare.

8. Documentarea codului

Pe parcursul implementării, elevul trebuie să adauge:

- comentarii explicative;
- note privind funcționarea algoritmilor;
- explicații pentru variabile complexe sau funcții esențiale.

O documentare bună este un indicator al profesionalismului și facilitează evaluarea lucrării.

9. Testarea finală și validarea funcționalităților

La final, aplicația este testată global:

- se verifică toate funcțiile;
- se testează fluxurile principale;
- se urmăresc scenarii speciale;
- se remediază ultimele erori.

Rezultatele testării sunt incluse în lucrare sub formă de capturi de ecran și descrieri.

Concluzie

Implementarea lucrării este un proces complex, dar bine structurat, în care elevul dovedește că stăpânește conceptele de programare și poate crea o aplicație funcțională, coerentă și eficientă. Această etapă reprezintă fundamentul evaluării competențelor profesionale în cadrul atestatului.

7. Testarea și validarea proiectului

Reprezintă etapa finală înainte de prezentarea lucrării de atestat și are rolul de a demonstra că aplicația funcționează corect, stabil și conform cerințelor stabilite. Această etapă confirmă faptul că soluția informatică este completă, fiabilă și îndeplinește obiectivele tehnice și funcționale prevăzute în planul de proiectare. Testarea și validarea nu se rezumă doar la găsirea erorilor, ci și la verificarea calității generale a aplicației.

Verificarea funcționalităților este esențială pentru o prezentare de succes.

Se recomandă:

- Testare pe scenarii multiple;
- Rezolvarea erorilor înainte de finalizare;
- Realizarea unui set de date demonstrative.

Profesorul coordonator poate solicita teste suplimentare sau demonstrații intermediare.

1. Importanța testării

Testarea permite:

- identificarea și corectarea erorilor înainte de prezentare;
- confirmarea funcționării corecte a tuturor modulelor;
- evaluarea stabilității, performanței și ergonomiei interfeței;
- validarea corectitudinii datelor procesate;
- demonstrarea profesionalismului în realizarea aplicației.

O aplicație netestată corespunzător poate conține erori ascunse, care afectează evaluarea finală.

2. Tipuri de testări utilizate în proiect

Elevul trebuie să realizeze mai multe tipuri de testări, în funcție de complexitatea aplicației:

a) Testare funcțională

Verifică dacă fiecare funcționalitate implementată îndeplinește cerințele definite.

Exemple:

- adăugarea unui utilizator;
- ștergerea sau modificarea datelor;
- generarea rapoartelor;
- calcularea rezultatelor.

Pentru fiecare funcție se construiesc scenarii de test cu date valide și invalide.

b) Testare integrată

Verifică modul în care modulele colaborează:

- interfața → logică → bază de date;
- modulul de calcul → modulul de afișare.

Această testare urmărește coerența întregului flux al aplicației.

c) Testare de validare

Confirmă că aplicația răspunde cerințelor inițiale stabilite în proiectare.

Este verificată lista de obiective:

- aplicația face ceea ce trebuie?
- sunt implementate toate funcțiile cerute?

d) Testare de securitate (unde este cazul)

Verifică:

- protejarea datelor introduse;
- prevenirea accesului neautorizat;
- izolarea funcțiilor sensibile.

e) Testare de performanță

Pentru aplicațiile mai complexe:

- testarea vitezei de răspuns;
- testarea volumului de date;
- analiza consumului de resurse.

3. Realizarea scenariilor de test

Elevul trebuie să creeze un set de scenarii și cazuri de test, pentru a acoperi:

- utilizarea normală;
- comportamentul aplicației la date incorecte;
- limite (ex.: câmpuri goale, valori negative, format greșit);
- situații excepționale (ex.: fișier inexistent, conexiune întreruptă).

Pentru fiecare test se notează:

- datele folosite;
- pașii urmați;
- rezultatul obținut;
- rezultatul așteptat.

4. Corectarea erorilor (debugging)

Pe baza testării, elevul:

- identifică erorile de logică;
- corectează problemele de afișare sau de structură;
- verifică din nou funcțiile afectate;
- optimizează porțiuni de cod dacă este necesar.

Este recomandat ca erorile descoperite să fie documentate în lucrare (optional), cu scurte note privind soluțiile implementate.

5. Validarea finală a proiectului

Validarea reprezintă confirmarea că:

- aplicația este complet funcțională;
- toate cerințele au fost îndeplinite;
- datele sunt procesate corect;
- interfața este utilizabilă și intuitivă;
- nu există erori critice.

Această etapă poate include feedback-ul profesorului coordonator sau al unor utilizatori reali (colegi, prieteni).

6. Documentarea rezultatelor testelor

În lucrare se includ:

- capturi de ecran pentru funcțiile testate;
- tabele cu scenarii de test și rezultat;
- observații privind corectitudinea funcțiilor.

Documentarea testării demonstrează rigoarea și profesionalismul elevului.

7. Concluzia testării

Prin testare și validare se garantează că aplicația:

- este complet funcțională;
- poate fi utilizată fără risc;
- îndeplinește obiectivele declarate;
- este suficient de robustă pentru a fi prezentată în fața comisiei.

8. Redactarea lucrării finale

Lucrarea trebuie să fie coerentă, tehnică și ușor de urmărit. Ea se redactează conform cerințelor liceului și se predă atât în format fizic, cât și digital.

Redactarea lucrării reprezintă etapa în care toate informațiile acumulate – documentarea teoretică, proiectarea aplicației, implementarea, testarea și validarea – sunt organizate într-o formă coerentă, clară și academică. Scopul acestei faze este de a prezenta proiectul într-un mod profesional, astfel încât comisia de evaluare să poată înțelege cu ușurință demersul, logica și rezultatele obținute.

Redactarea corectă este esențială deoarece lucrarea scrisă reprezintă cartea de vizită a elevului și completează partea practică prin explicații, justificări și analize.

Structură recomandată:

1. Introducere
2. Capitol teoretic
3. Descrierea aplicației
4. Capturi de ecran și explicații
5. Concluzii
6. Bibliografie
7. Anexe (cod sursă, diagrame, planuri)

1. Structurarea coerentă a conținutului

Lucrarea finală trebuie organizată într-o structură logică, bine definită. Cele mai utilizate capitole sunt:

1. **Introducere**
 - prezentarea temei;
 - motivația alegerii acesteia;
 - obiectivele lucrării;
 - metodologia de realizare.
2. **Cadrul teoretic**
 - noțiuni fundamentale necesare înțelegerii aplicației;
 - concepte, algoritmi, limbaje de programare;
 - tehnologii utilizate;
 - exemple și referințe relevante.

3. Proiectarea aplicației

- diagrame UML;
- descrierea arhitecturii;
- prezentarea modulelor;
- schema bazei de date.

4. Implementarea

- prezentarea codului reprezentativ;
- explicarea funcțiilor principale;
- capturi de ecran;
- instrumentele folosite.

5. Testarea și validarea

- scenarii de test;
- tabele cu rezultate;
- erori identificate și remediate.

6. Concluzii

- evaluarea rezultatelor;
- limitele aplicației;
- direcții de dezvoltare ulterioară.

7. Bibliografie

- surse de informare, articole, manuale, documentații oficiale.

8. Anexe (opțional)

- cod integral;
- tabele extinse;
- alte materiale suplimentare.

2. Respectarea normelor academice

Redactarea trebuie să respecte reguli de stil și prezentare:

- limbaj formal, clar și concis;
- propoziții bine structurate;
- evitarea exprimărilor vagi sau incomplete;
- utilizarea corectă a terminologiei de specialitate;
- respectarea normelor ortografice și gramaticale;

- utilizarea corectă a citatelor (unde este cazul).

Lucrarea trebuie să fie lizibilă, coerentă și lipsită de erori.

3. Utilizarea elementelor vizuale

Elementele grafice contribuie la înțelegerea materialului:

- capturi de ecran ale aplicației;
- diagrame de clase, activități, secvențe;
- schema bazei de date;
- grafice sau tabele pentru prezentarea rezultatelor;
- fragmente de cod evidențiate.

Aceste elemente trebuie numerotate și descrise clar.

4. Explicarea logică a codului și funcționalităților

Codul nu trebuie prezentat doar ca text, ci explicat:

- ce face funcția respectivă;
- cum contribuie la funcționarea aplicației;
- de ce a fost aleasă o anumită soluție tehnică;
- ce tehnologii sunt folosite și care sunt avantajele lor.

Explicațiile trebuie să demonstreze înțelegerea profundă a mecanismelor interne.

5. Claritatea argumentării și justificării

Lucrarea trebuie să conțină:

- justificări ale alegerilor tehnologice;
- argumente pentru structura proiectului;
- motive pentru implementarea funcțiilor în forma prezentată;
- analiza avantajelor și limitelor.

Acest tip de argumentare arată maturitate profesională.

6. Corectura și revizia finală

Înainte de predare, elevul trebuie să:

- recitească lucrarea integral;
- corecteze eventualele greșeli;
- verifice consistența numerotării figurilor și tabelor;
- verifice corectitudinea bibliografiei;
- asigure o formatare uniformă (font, alineate, titluri).

Este util să fie implicat și profesorul coordonator pentru o ultimă revizuire.

7. Pregătirea versiunii finale pentru predare

Versiunea finală trebuie să fie:

- corectă, completă și bine structurată;
- tipărită cu grijă (dacă este cerut);
- însoțită de suportul digital al aplicației (stick / CD / online).

Fișierele trebuie organizate într-un mod profesionist:

- codul sursă;
- documentația;
- baza de date;
- versiunea executabilă (dacă este cazul).

Concluzie

Redactarea lucrării finale este procesul prin care proiectul devine un document academic, capabil să reflecte nivelul de pregătire profesională al elevului. O lucrare bine redactată demonstrează nu doar competențe tehnice, ci și abilități de comunicare, organizare și analiză – esențiale pentru orice viitor specialist în domeniul informaticii.

9. Prezentarea lucrării în fața comisiei

Este etapa finală a procesului de certificare a competențelor profesionale și reprezintă momentul în care elevul demonstrează nu doar că stăpânește tehnic lucrarea realizată, ci și că poate comunica coerent, structurat și profesionist rezultatele obținute. O prezentare bine pregătită poate face diferența între o evaluare bună și una excelentă, chiar dacă lucrarea tehnică este aceeași.

Prezentarea trebuie să fie:

- Clară și bine structurată;
- În intervalul de timp alocat (de obicei 5–10 minute);
- Suportată de demonstrarea practică a aplicației;
- Însotită de o explicație logică a alegerilor tehnice.

1. Pregătirea generală a prezentării

Înainte de susținerea orală, elevul trebuie să pregătească o prezentare clară, de regulă sub forma unui slideshow (PowerPoint, Google Slides etc.), care să conțină elementele esențiale ale lucrării:

- titlul proiectului și datele de identificare ale elevului;
- obiectivele aplicației;
- tehnologiile utilizate;
- structura generală a aplicației;
- pașii principali în realizarea proiectului;
- capturi de ecran relevante;
- funcționalitățile cheie;
- concluziile și posibilitățile de dezvoltare ulterioară.

Prezentarea trebuie să fie concisă, vizuală și ușor de urmărit, evitând textul prea dens.

2. Structurarea discursului oral

Discursul elevului trebuie să urmeze o structură logică și profesională:

a) Introducere

- prezentarea numelui, a temei și a scopului proiectului;
- motivarea alegerii temei;
- importanța practică a aplicației dezvoltate.

b) Conținutul lucrării

Elevul prezintă pe scurt:

- descrierea cadrului teoretic folosit;
- arhitectura generală a proiectului;
- principalele module și logica lor;
- abordarea adoptată în proiectare și implementare.

c) Demonstrarea aplicației

Aceasta este cea mai importantă parte.

Elevul trebuie să arate comisiei:

- modul de pornire a aplicației;
- interfața utilizatorului;
- funcțiile esențiale (adăugare, editare, căutare, generare rapoarte etc.);
- situații speciale (validări, erori, mesaje informative).

Demonstrarea trebuie să fie fluentă, fără ezitări, explicând clar fiecare pas.

d) Rezultate și testări

Elevul prezintă:

- modul în care a testat aplicația;
- scenarii importante de test;
- eventualele optimizări făcute după identificarea unor probleme.

e) Concluzii

- evaluarea reușitei proiectului;
- competențele dezvoltate;
- limitele aplicației;
- idei de extindere ulterioară.

3. Gestionarea timpului

Prezentarea trebuie încadrată în timpul oferit (de regulă 5–10 minute).

Elevul trebuie să:

- exerseze prezentarea de mai multe ori;
- elimine detaliile prea tehnice;
- se concentreze pe ideile principale, pe logica proiectului și pe demonstrarea practică.

4. Atitudinea în fața comisiei

Comportamentul elevului transmite profesionalism:

- ton calm, clar și sigur;
- privirea către membrii comisiei, nu doar către ecran;
- ritm potrivit al vorbirii;
- postură deschisă, fără gesturi excesive;
- disponibilitatea de a răspunde la întrebări.

Este important ca elevul să arate că înțelege lucrarea și nu doar că a memorat textul prezentării.

5. Gestionarea întrebărilor

După prezentare, comisia poate adresa întrebări legate de:

- algoritmi folosiți;
- justificarea soluțiilor tehnice;
- funcționalitatea aplicației în anumite scenarii;
- limite sau probleme specifice;
- noțiuni teoretice folosite în lucrare.

Elevul trebuie să:

- răspundă calm și respectuos;
- recunoască dacă nu cunoaște un detaliu, fără a inventa răspunsuri;
- argumenteze deciziile tehnice cu logică și claritate;
- demonstreze înțelegerea profundă a proiectului.

6. Aspectul tehnic al prezentării

Pentru ca prezentarea să decurgă fără probleme:

- aplicația trebuie testată înainte de susținere;
- fișierele necesare (proiectul, baza de date, prezentarea PowerPoint) să fie pregătite într-un folder organizat;
- elevul să aibă o copie de rezervă pe un stick sau în cloud;
- eventualele setări software necesare să fie verificate înainte.

7. Impactul prezentării asupra notei finale

O prezentare bine susținută:

- demonstrează autonomie și maturitate;
- clarifică modul de gândire al elevului;
- completează partea tehnică a proiectului;

- impresionează comisia prin profesionalism.

Chiar și o lucrare medie tehnic poate primi o notă bună dacă prezentarea este excelentă, iar o lucrare bună poate fi valorificată maxim printr-o susținere clară și coerentă.

10. Rolul profesorului coordonator

Profesorul are responsabilități esențiale:

- Ghidarea elevului în alegerea temei;
- Supravegherea progresului;
- Oferirea de feedback tehnic;
- Verificarea originalității lucrării;
- Sprijinirea elevului în pregătirea prezentării.

Este esențial în întregul proces de realizare a lucrării de atestat, deoarece acesta ghidează elevul, îl sprijină metodologic și tehnic și asigură ca proiectul respectă standardele academice și profesionale cerute în certificarea competențelor. Profesorul coordonator nu este doar un evaluator, ci un mentor, un consultant și un garant al calității.

1. Orientarea elevului în alegerea temei

Profesorul ajută elevul:

- să identifice o temă potrivită nivelului său de pregătire;
- să aleagă un subiect realizabil în timpul disponibil;
- să înțeleagă relevanța și aplicabilitatea temei;
- să evite proiectele prea complicate sau prea simple.

Coordonatorul poate sugera teme, direcții sau îmbunătățiri, asigurând echilibrul între interesul elevului și cerințele examenului.

2. Stabilirea obiectivelor și a planului de lucru

Profesorul coordonator:

- ajută la formularea clară a obiectivelor lucrării;
- definește împreună cu elevul etapele de lucru;
- stabilește termene pentru predarea parțială a materialelor;
- monitorizează progresul și reorientează activitatea dacă e necesar.

Acest rol de planificare previne întârzierile și blocajele în implementare.

3. Oferirea de suport teoretic și tehnic

Pe parcursul realizării proiectului, profesorul:

- explică noțiuni teoretice necesare temei;
- clarifică algoritmi, structuri de date și tehnologii utilizate;
- sugerează resurse bibliografice sau tutoriale;

- oferă recomandări tehnice pentru optimizarea codului.

Rolul său nu este de a scrie codul, ci de a orienta elevul către soluțiile corecte.

4. Monitorizarea progresului și corectarea direcției de lucru

Profesorul coordonator:

- urmărește evoluția proiectului prin întâlniri regulate;
- analizează secțiuni de cod, diagrame, capitole parțiale;
- semnalează probleme de logică, organizare sau documentare;
- propune soluții pentru rezolvarea dificultăților.

Această monitorizare continuă asigură calitatea proiectului și respectarea cerințelor.

5. Revizuirea documentației finale

Profesorul joacă un rol major în redactare:

- corectează forma și conținutul lucrării scrise;
- verifică exactitatea explicațiilor tehnice;
- se asigură că lucrarea are o structură logică și completă;
- oferă feedback pentru îmbunătățirea clarității și acurateței informațiilor.

Scopul său este ca lucrarea să fie corectă, coerentă și profesional redactată.

6. Evaluarea corectitudinii implementării

Coordonatorul analizează:

- funcționalitățile aplicației realizate de elev;
- calitatea codului;
- modul de respectare a proiectării;
- robustețea și claritatea soluțiilor implementate.

Profesorul îl ajută pe elev să identifice și să rezolve erorile înainte de prezentarea finală.

7. Pregătirea elevului pentru prezentare

Profesorul:

- oferă recomandări privind structura prezentării orale;
- sugerează elementele de evidențiat în timpul demonstrării aplicației;
- simulează posibile întrebări ale comisiei;
- antrenează elevul să vorbească clar, concis și încrezător.

Astfel, elevul ajunge la momentul susținerii bine pregătit și sigur pe sine.

8. Asigurarea respectării standardelor examenului

Printre responsabilitățile sale se numără:

- verificarea originalității lucrării;
- respectarea cerințelor oficiale (format, conținut, structură);
- garantarea faptului că proiectul reprezintă munca reală a elevului;
- completarea documentelor administrative necesare.

Profesorul coordonator este responsabil direct pentru integritatea proiectului prezentat.

9. Oferirea de feedback constructiv și sprijin moral

Pe lângă rolul tehnic, profesorul:

- încurajează elevul;
- îi oferă încredere în propriile abilități;
- îl motivează să depășească dificultățile;
- îl ajută să-și gestioneze emoțiile înainte de prezentare.

Acest rol de mentor are un impact major asupra reușitei elevului.

Concluzie

Rolul profesorului coordonator este complex, combinând îndrumare tehnică, suport teoretic, monitorizare, evaluare și mentorat. Prezența sa activă și competență contribuie decisiv la succesul elevului, încurajându-l să dezvolte un proiect solid, original și bine prezentat, care să reflecte competențele dobândite în profilul Matematică–Informatică.

11. Etică și originalitate

Lucrarea trebuie să fie originală și să reflecte munca elevului.

Este obligatoriu:

- Să se evite plagiatul;
- Să se citeze corect sursele;
- Să se menționeze contribuțiile personale;
- Să se respecte regulile de etică academică.

Etica și originalitate reprezintă două principii fundamentale în realizarea unei lucrări de atestat la profilul Matematică–Informatică, deoarece acestea garantează nu doar corectitudinea procesului de învățare, ci și integritatea profesională a elevului. Respectarea lor demonstrează maturitate, responsabilitate și capacitatea de a produce conținut propriu, competențe esențiale în orice domeniu tehnic.

1. Importanța originalității în realizarea lucrării

Originalitatea este un criteriu esențial al evaluării lucrării de atestat. Ea nu se referă neapărat la inventarea unui concept complet nou, ci la modul în care elevul:

- formulează ideile cu propriile cuvinte;
- structurează explicațiile;
- implementează codul;
- propune soluții tehnice adaptate temei;
- construiește aplicația într-o manieră personală.

O lucrare originală:

- evidențiază înțelegerea profundă a subiectului;
- dovedește că elevul a lucrat efectiv la proiect;
- conferă credibilitate prezentării în fața comisiei;
- evită orice suspiciune de plagiat.

În contextul informaticii, originalitatea se reflectă în:

- modul de structurare al proiectului;
- algoritmi aleși;
- interfața aplicației;
- organizarea codului și documentarea internă;
- soluțiile proprii la probleme tehnice.

2. Respectarea eticii academice

Etica în elaborarea lucrării include:

- recunoașterea corectă a surselor utilizate;
- asumarea responsabilității pentru codul scris;
- evitarea copiatului și a reutilizării nepermise de materiale;
- respectarea indicațiilor profesorului coordonator;
- prezentarea corectă și onestă a contribuției proprii.

Este normal și acceptat ca elevul:

- să consulte manuale, tutoriale, forumuri și cod deschis;
- să se inspire din proiecte similare;
- să folosească exemple de cod cu licență liberă.

Însă, pentru a respecta etica academică, este obligatoriu:

- să precizeze sursele (bibliografie, linkuri, autori);
- să adapteze și să explice orice fragment utilizat;
- să nu includă în lucrare secțiuni copiate integral fără citare;
- să nu prezinte drept propriu un proiect realizat de altcineva.

Scopul examenului este verificarea competențelor elevului, nu reproducerea unei soluții găsite online.

3. Gestionarea corectă a surselor de documentare

Pentru a menține standardele de etică, elevul trebuie:

- să includă bibliografia la finalul lucrării;
- să specifice clar sursele pentru imagini, diagrame, teorii sau cod;
- să evite citatele lungi și să folosească parafrizarea;
- să menționeze licențele software utilizate (open-source sau comerciale).

Astfel, lucrarea devine transparentă, bine documentată și credibilă.

4. Originalitate în partea practică (cod și aplicație)

Codul este o componentă în care originalitatea contează foarte mult. Elevul trebuie:

- să scrie personal majoritatea funcțiilor;
- să înțeleagă fiecare linie de cod pe care o include;
- să nu folosească proiecte gata făcute descărcate de pe internet;
- să nu recurgă la generatoare automate fără să explice rezultatul.

Este permisă utilizarea:

- funcțiilor standard;
- bibliotecilor oficiale;
- framework-urilor;
- modulelor cu licență liberă.

Totuși, implementarea trebuie să fie personalizată și adaptată temei.

5. Consecințele nerespectării eticii și originalității

Plagiatul sau prezentarea unei lucrări realizate de altcineva poate duce la:

- respingerea lucrării;
- anularea evaluării;
- scăderea notei;
- repetarea examenului de certificare;
- pierderea credibilității în fața comisiei.

În unele cazuri grave, școala poate aplica și sancțiuni disciplinare.

Elevii trebuie să înțeleagă că atestatul este nu doar o evaluare, ci și un exercițiu de responsabilitate profesională.

6. Beneficiile respectării eticii și originalității

Atunci când lucrarea este realizată corect:

- elevul învață cu adevărat;
- își dezvoltă gândirea critică și abilitățile tehnice;
- își poate prezenta proiectul cu încredere;
- obține o notă care reflectă munca depusă;
- dobândește experiență utilă pentru examene și proiecte viitoare.

Respectarea eticii este un pas important spre maturitatea profesională.

12. Exemple de teme pentru lucrări de atestat

Aplicații software

- Gestionarea unei biblioteci (C++/MySQL)
- Sistem de evidență a elevilor (Java/Python)
- Agenda personală cu interfață grafică (Python – Tkinter)
- Soft educațional: exerciții de matematică pentru gimnaziu
- Aplicație de gestiune a unui magazin (PHP/SQL)

Jocuri și simulări

- Joc tip „Spânzurătoarea”
- Simulare trafic rutier
- Labirint 2D cu algoritmi de pathfinding
- Joc de strategie simplă cu interfață grafică

Aplicații web

- Blog personal cu panou de administrare
- Magazin online simplificat
- Website pentru un club școlar
- Sistem de programări online

Modelare matematică și algoritmică

- Generarea fractalilor (Mandelbrot, Julia)
- Criptografie: implementarea algoritmului RSA
- Algoritmi de sortare – comparație și vizualizare
- Optimizare prin metode numerice

Elevii vor folosi și aplicațiile învățate în urma participării la proiectul **Transformarea Educației Militare prin Competențe Informatică Avansate**

13. Exemple de plan de lucru (model)

1. **Săptămâna 1–2:** Alegerea temei și aprobarea acesteia.
2. **Săptămâna 3–4:** Documentare teoretică și strângerea surselor.
3. **Săptămâna 5–6:** Proiectarea aplicației (diagrame, baze de date, interfață).
4. **Săptămâna 7–10:** Implementare etapizată a funcționalităților principale.
5. **Săptămâna 11–12:** Testare și corectarea erorilor.
6. **Săptămâna 13:** Redactarea lucrării scrise.
7. **Săptămâna 14:** Realizarea prezentării.
8. **Săptămâna 15:** Simulare de susținere și ajustări.

14. Erori frecvente și cum pot fi evitate

Realizarea lucrării de atestat este un proces complex, care combină documentarea, proiectarea, implementarea, testarea și prezentarea finală. De aceea, este important ca elevii să fie conștienți de capcanele și erorile frecvent întâlnite, care pot afecta calitatea lucrării sau nota finală. În continuare sunt prezentate cele mai comune greșeli, alături de recomandări pentru a le evita.

- **Tema prea complicată:** alege o temă care se poate realiza în timpul disponibil.
- **Lipsa planificării:** folosește un calendar și urmărește termenele.
- **Documentare insuficientă:** include cel puțin 5–10 surse credibile.
- **Cod nestructurat:** comentează funcțiile și folosește convenții de numire.
- **Neatingerea cerințelor minime:** verifică mereu cu profesorul coordonator.
- **Prezentare nepregătită:** repetă în avans discursul și testul aplicației.

1. Alegerea unei teme prea generale sau prea complicate

O temă prea vastă face imposibilă redactarea unei lucrări coerente, iar una prea tehnică depășește nivelul de pregătire al elevului. Ambele situații duc la dificultăți majore în implementare și în prezentarea finală.

Cum eviți această greșală:

- Alege o temă specifică, bine delimitată și adaptată competențelor tale.
- Consultă profesorul coordonator înainte de alegere.
- Verifică dacă poți găsi resurse suficiente și dacă ai timpul necesar pentru implementarea practică.

2. Documentarea superficială sau neorganizată

O lucrare fără bază teoretică solidă este incompletă și transmite lipsa de profesionalism.

De asemenea, documentarea făcută haotic duce la explicații neclare și inconsecvente.

Cum eviți această greșală:

- Folosește surse variate și credibile.
- Notează informațiile pe măsură ce le studiezi.
- Construiește partea teoretică în ordinea logică a conceptelor.

3. Copierea nejustificată a informațiilor sau a codului (plagiatul)

Plagiatul este una dintre cele mai grave greșeli, ducând la respingerea lucrării, anularea examenului sau alte sancțiuni disciplinare.

Cum eviți această greșală:

- Scrie explicațiile cu propriile cuvinte.
- Citează orice sursă utilizată.
- Nu copia proiecte gata făcute; implementează soluții proprii.

4. Lipsa unui plan de lucru și a organizării pe etape

Multe lucrări devin dificile nu din cauza complexității temei, ci a lipsei unui plan clar. Elevii tind să lucreze haotic sau prea târziu.

Cum eviți această greșală:

- Creează un calendar realist de activități.
- Stabilește obiective săptămânale.
- Respectă etapele: documentare → proiectare → implementare → testare → redactare.

5. Ignorarea proiectării și trecerea directă la cod

Una dintre cele mai frecvente erori în informatică este scrierea codului fără:

- diagrama aplicației;
- definirea structurilor de date;
- planificarea funcțiilor sau modulelor.

Rezultatul este un cod dificil de înțeles și plin de erori.

Cum eviți această greșală:

- Elaborează mai întâi schema logică a programului.
- Realizează schițele interfeței.
- Planifică organizarea folderelor și a fișierelor codului.

6. Lipsa testării sau testarea insuficientă a aplicației

O aplicație care rulează doar în anumite condiții este considerată incompletă. Testarea superficială poate ascunde erori care apar în timpul prezentării în fața comisiei.

Cum eviți această greșeală:

- Testează fiecare funcție separat.
- Simulează scenarii de utilizare diferite.
- Verifică erorile de tip input invalid, crash-uri sau rezultate incorecte.

7. Redactarea neglijentă a documentului final

Erorile de ortografie, paginile nealiniate, explicațiile incomplete sau lipsa coerenței afectează impresia generală.

Cum eviți această greșeală:

- Folosește un stil clar, academic.
- Revizuieste lucrarea de mai multe ori.
- Integrează ilustrări relevante (capturi de ecran, diagrame, exemple de cod).

8. Prezentarea nepregătită în fața comisiei

Mulți elevi greșesc crezând că aplicația și lucrarea „vor vorbi singure”. În realitate, evaluarea depinde în mare măsură de prezentare.

Cum eviți această greșeală:

- Exersează prezentarea de câteva ori.
- Familiarizează-te cu toate funcțiile aplicației.
- Răspunde clar și calm la întrebările comisiei.

9. Dependența excesivă de profesor sau de resurse externe

A cere ajutor este firesc, însă dependența totală duce la lipsa înțelegerii profunde a soluției.

Cum eviți această greșeală:

- Rezolvă singur majoritatea problemelor.
- Folosește sugestiile profesorului ca direcții, nu ca soluții complete.
- Caută soluții alternative înainte de a cere ajutor.

10. Neglijarea cerințelor formale

Lucrarea trebuie să respecte strict:

- formatul stabilit de școală;
- numărul de pagini;
- structura obligatorie;
- normele de citare;
- termenul de predare.

Cum eviți această greșeală:

- Citește cerințele oficiale încă de la început.
- Verifică lucrarea înainte de predare.
- Întreabă profesorul coordonator dacă ai nelămuriri.

15. Checklist final pentru elevi

- Am ales o temă relevantă și aprobată.
- Am realizat partea teoretică și am citat corect sursele.
- Am proiectat aplicația înainte de a scrie codul.
- Am implementat funcționalitățile minimale și opționale.
- Am testat aplicația și am eliminat erorile.
- Am realizat capturi de ecran și explicații pentru lucrarea scrisă.
- Am completat bibliografia și anexele.
- Am pregătit discursul pentru comisie.

16. Checklist pentru profesorul coordonator

Aprobarea temei și nivelului de dificultate.

- Revizuirea etapelor de progres.
- Sugestii pentru îmbunătățirea designului și a algoritmilor.
- Verificarea originalității lucrării.
- Validarea versiunii finale.
- Pregătirea elevului pentru prezentare.

17. Anexe utile

Formular pentru jurnal de progres

- Data:
- Activități realizate:
- Dificultăți întâmpinate:
- Soluții aplicate:
- Planul pentru următoarea întâlnire:

Model pentru bibliografie

- Autor, *Titlu*, Editura, Anul.
- URL complet pentru sursele online.

Model prezentare PowerPoint

1. Titlul proiectului
2. Obiective
3. Fundamente teoretice

4. Descrierea aplicației
5. Capturi și demonstrație
6. Concluzii
7. Mulțumiri

18. Concluzii

Realizarea lucrării de certificare a competențelor profesionale (atestat) reprezintă un proces complex, dar esențial în formarea elevilor de la profilul Matematică–Informatică. Întregul demers — de la alegerea temei, documentare și proiectare, până la implementare, testare, redactare și prezentarea finală — contribuie la dezvoltarea unui set solid de competențe tehnice, academice și profesionale, care depășesc cadrul strict al examenului.

În primul rând, acest proiect oferă oportunitatea de a aplica în mod practic cunoștințele dobândite pe parcursul anilor de liceu. Elevii ajung să lucreze cu instrumente, limbaje de programare, concepte matematice și metode de proiectare pe care le-au studiat teoretic, transformându-le în soluții concrete la probleme reale. Astfel, proiectul devine un liant între teoria acumulată și practica necesară în domeniul informaticii.

În al doilea rând, realizarea unei lucrări de atestat contribuie la formarea unei mentalități orientate către proiect, învățând elevii să planifice, să organizeze și să ducă la bun sfârșit un obiectiv complex. Ei își dezvoltă abilități critice precum: gestionarea timpului, rezolvarea problemelor, lucrul structurat pe etape, documentarea coerentă și verificarea calității propriei munci.

Dimensiunea academică a lucrării este, de asemenea, importantă. Elevii învață să redacteze un material teoretic coerent, bine argumentat, respectând normele de redactare și etică profesională. În acest proces, se obișnuiesc cu responsabilitatea intelectuală, înțelegând importanța citării surselor, a originalității și a respectării integrității academice.

Mai mult, componenta practică a proiectului stimulează creativitatea și inovația, oferind elevilor libertatea de a experimenta cu noi tehnologii, interfețe, algoritmi sau strategii de implementare. În acest fel, lucrarea de atestat poate deveni un prim portofoliu profesional, valoros pentru viitoare admitere la facultate sau pentru oportunități de angajare.

Prezentarea lucrării în fața comisiei reprezintă un moment formativ în sine, în care elevul își dezvoltă abilitățile de comunicare, învățând să explice clar, concis și profesionist un proiect tehnic. Această experiență contribuie la consolidarea încrederii în sine și a capacității de autoprezentare, competențe esențiale în mediile academice și profesionale.

Nu în ultimul rând, rolul profesorului coordonator contribuie la formarea elevului ca mic cercetător, oferind ghidaj, feedback și suport, dar încurajând autonomia și gândirea critică.

În ansamblu, lucrarea de atestat nu este doar un examen, ci o experiență educațională

completă, care îi pregătește pe elevi pentru provocările următoare: studiile universitare, proiectele viitoare și profesiile din domeniul tehnic. Ea sintetizează cunoștințele unui ciclu de studiu, dezvoltă competențe esențiale și stimulează o atitudine responsabilă, creativă și orientată spre performanță.

Un management eficient al lucrării de competențe profesionale contribuie la succesul elevilor, la dezvoltarea competențelor tehnologice și la consolidarea relației dintre profesor și elev. Prin planificare, disciplină, creativitate și colaborare, lucrarea de atestat devine o experiență profesională valoroasă și o etapă importantă în drumul către studiile superioare și carieră.

19. BIBLIOGRAFIE

1. M E T O D O L O G I A de organizare și desfășurare a examenului de atestare a competențelor profesionale a absolvenților claselor de matematică-informatică și matematică-informatică, intensiv informatică, Ordinul ministrului educației, cercetării și inovării nr. 4.843/2009.
2. ORDIN Nr.3009/2021 pentru modificarea și completarea Metodologiei de organizare și desfășurare a examenului de atestare a competențelor profesionale ale absolvenților claselor de matematică-informatică și matematică-informatică, intensiv informatică, aprobată prin Ordinul ministrului educației, cercetării și inovării nr. 4.843/2009.
3. Anexa la Ordinul MEN nr. 4433/ 29.08.2014 privind aprobarea Metodologiei de organizare și desfășurare a examenului de certificare a calificării absolvenților învățământului liceal, filiera vocațională, cu modificările și completările ulterioare.

GRAB YOUR READER'S ATTENTION WITH A GREAT
QUOTE FROM THE DOCUMENT OR USE THIS
SPACE TO EMPHASIZE A KEY POINT. TO PLACE THIS
TEXT BOX ANYWHERE ON THE PAGE, JUST DRAG
IT!

**MANAGEMENTUL LUCRĂRILOR DE CERTIFICARE A
COMPETENȚELOR PROFESIONALE (ATESTAT) LA PROFILUL
MATEMATICĂ-INFORMATICĂ
ISBN 978-973-0-43318-0**