# Aplicatie pentru evaluarea rezistentei si stabilitatii structurilor in constructii

# Documentul de proiectare

Cuprins

[1. Introducere 1](#_Toc160527836)

[1.1 Scopul documentului 1](#_Toc160527837)

[2. Prezentare generală și abordări de proiectare 2](#_Toc160527838)

[2.1 Prezentare generală 2](#_Toc160527839)

[2.2 Presupuneri/ Constrângeri/ Riscuri 2](#_Toc160527840)

[2.2.1 Presupuneri 2](#_Toc160527841)

[2.2.2 Constrângeri 2](#_Toc160527842)

[2.2.3 Riscuri 3](#_Toc160527843)

[3. Considerațiii de proiectare 4](#_Toc160527844)

[3.1 Obiective și linii directoare (ghiduri) 4](#_Toc160527845)

[3.2 Metode de dezvoltare 4](#_Toc160527846)

[3.3 Strategii de arhitectură 4](#_Toc160527847)

[4. Arhitectura Sistemului și Proiectarea Arhitecturii 6](#_Toc160527848)

[4.1 Vedere logică 6](#_Toc160527849)

[4.2 Arhitectură hardware 6](#_Toc160527850)

[4.3 Arhitectură software 6](#_Toc160527851)

[4.4 Arhitectura informațiilor 7](#_Toc160527852)

[4.5 Arhitectura de comunicații interne 7](#_Toc160527853)

[4.6 Diagrama de arhitectură a sistemului 8](#_Toc160527854)

[5. Proiectarea sistemului 9](#_Toc160527855)

[5.1 Proiectarea bazei de date 9](#_Toc160527856)

[5.1.1 Obiecte de date și structuri de date rezultante 9](#_Toc160527857)

[5.1.2 Fișiere și baze de date 9](#_Toc160527858)

[5.2 Conversii de date 9](#_Toc160527859)

[5.3 Interfețe utilizator 10](#_Toc160527860)

[5.3.1 Intrări 10](#_Toc160527861)

[5.3.2 Ieșiri 10](#_Toc160527862)

[5.4 Proiectarea interfețelor cu utilizatorul 10](#_Toc160527863)

[6. Scenarii de utilizare 11](#_Toc160527864)

[7. Proiectare de detaliu 12](#_Toc160527865)

[7.1 Proiectare hardware de detaliu 12](#_Toc160527866)

[7.2 Proiectare software de deatliu 12](#_Toc160527867)

[7.3 Proiectare detaliată de securitate 13](#_Toc160527868)

[7.4 Proiectare de detaliu pentru performanța sistemului 14](#_Toc160527869)

[7.5 Proiectare detaliată a comunicațiilor interne (între componente) 14](#_Toc160527870)

[8. Controale pentru verificarea integrității sistemului 15](#_Toc160527871)

[Anexa A: Gestiunea modificărilor documentului 16](#_Toc160527872)

[Anexa B: Acronime 17](#_Toc160527873)

[Anexa C Documente la care se face referire 18](#_Toc160527874)

## Introducere

Instrucțiuni: Furnizați informații de identificare pentru sistemul existent și/sau propus sau situația pentru care se aplică Documentul de Proiectare a Sistemului (de exemplu, numele complet și acronimele proiectului de dezvoltare, sistemul sau situația existentă și propusă, după caz), și evoluția așteptată a documentului. De asemenea, descrieți orice considerații referitor la securitate sau confidențialitate asociate cu utilizarea acestui document..

Documentul de Proiectare a Sistemului descrie modul în care cerințele funcționale și non-funcționale înregistrate în Documentul de Cerințe se transformă în specificații de proiectare a sistemului, mai tehnice, pe baza cărora se construiește sistemul. Se documentează proiectarea sistemului la un nivel înalt și specificațiile detaliate de proiectare.

Se descriu obiectivele și considerațiile de proiectare, se furnizează o prezentare generală la nivel înalt a arhitecturii sistemului și se descrie proiectarea datelor asociate cu sistemul, precum și interfața om-mașină și scenariile operaționale.

Proiectarea sistemului la nivel înalt este descompusă în continuare în specificații detaliate de proiectare pentru fiecare componentă a sistemului, inclusiv hardware, comunicațiile interne, software, controalele de integritate a sistemului și interfețele externe.

### Scopul documentului

Acest document de proiectare detaliază arhitectura și proiectarea aplicației 'Aplicație pentru evaluarea rezistenței și stabilității structurilor în construcții'. El descrie în detaliu modul în care cerințele funcționale și non-funcționale, specificate în Documentul de Specificare a Cerințelor (SRS), sunt transformate în specificații de proiectare tehnice și implementabile. Scopul principal al acestui document este de a oferi o bază solidă pentru echipa de dezvoltare, furnizând îndrumări clare și precise asupra arhitecturii sistemului, componentelor software și hardware, interfețelor cu utilizatorul, structurii bazei de date și a altor aspecte tehnice cruciale. Documentul acoperă atât aspectele de nivel înalt ale arhitecturii sistemului, oferind o viziune generală asupra componentelor principale și a interacțiunilor dintre ele, cât și specificațiile detaliate ale fiecărei componente în parte, asigurând astfel că echipa de dezvoltare are toate informațiile necesare pentru a construi aplicația conform cerințelor stabilite.

## Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile care vor fi utilizate ca ghiduri în momentul proiectării și implementării sistemului.

### Prezentare generală

Aplicația va fi proiectată ca o aplicație desktop, adoptând o arhitectură stratificată. Această arhitectură va separa clar diferitele preocupări ale aplicației (prezentare, logică de afaceri, acces la date), ceea ce va facilita mentenabilitatea, scalabilitatea și testarea. Interfața cu utilizatorul va fi dezvoltată folosind biblioteca PyQt, care oferă un set bogat de instrumente și componente pentru crearea de interfețe grafice moderne și interactive. Logica de calcul, care include algoritmii și formulele pentru evaluarea rezistenței și stabilității structurilor, va fi implementată în limbajul de programare Python, ales pentru claritatea, flexibilitatea și ecosistemul său vast de biblioteci științifice. Datele, inclusiv proprietățile materialelor, datele de intrare ale utilizatorului și rezultatele calculelor, vor fi stocate local într-o bază de date SQLite. SQLite a fost selectat datorită simplității sale, a amprentei reduse și a faptului că nu necesită un server separat, ceea ce simplifică implementarea și distribuția aplicației.

### Presupuneri/ Constrângeri/ Riscuri

#### Presupuneri

Dezvoltarea aplicației se va baza pe anumite presupuneri. În primul rând, se presupune că utilizatorii aplicației posedă cunoștințe de bază în domeniul calculului structural, fiind familiarizați cu conceptele și terminologia specifică. De asemenea, aplicația este proiectată pentru a fi utilizată în principal pe computere cu sistem de operare Windows, ceea ce influențează alegerile tehnologice și de proiectare.

#### Constrângeri

Proiectarea și implementarea aplicației sunt supuse anumitor constrângeri. O constrângere importantă este cerința ca aplicația să funcționeze offline, fără a necesita o conexiune activă la internet. Această constrângere implică faptul că toate datele necesare pentru funcționarea aplicației vor fi stocate local. O altă constrângere este legată de tehnologia utilizată pentru dezvoltarea interfeței grafice, care va fi realizată exclusiv cu ajutorul bibliotecii PyQt.

#### Riscuri

Dezvoltarea aplicației implică anumite riscuri care trebuie identificate și gestionate. Un risc potențial este reprezentat de posibilele evoluții viitoare ale normativelor în construcții, care ar putea necesita modificări ale aplicației pentru a asigura conformitatea. Pentru a reduce acest risc, aplicația va fi proiectată modular, ceea ce va facilita actualizările și adaptările. Un alt risc este legat de complexitatea calculelor structurale, care ar putea afecta performanța aplicației, în special pentru calcule complexe sau pe computere cu resurse limitate. Pentru a atenua acest risc, se vor optimiza algoritmii de calcul și se vor stabili cerințe hardware minime pentru a asigura o experiență de utilizare acceptabilă.

## Considerații de proiectare

### Obiective și linii directoare (ghiduri)

Proiectarea aplicației va fi ghidată de o serie de obiective și linii directoare clare, menite să asigure calitatea și eficiența produsului final. Un obiectiv primordial este ușurința în utilizare, ceea ce implică dezvoltarea unei interfețe cu utilizatorul intuitive și ușor de navigat, chiar și pentru utilizatorii care nu au experiență extinsă cu aplicații de calcul structural. Un alt obiectiv fundamental este acuratețea, deoarece calculele structurale efectuate de aplicație trebuie să fie precise și să respecte cu strictețe normativele în vigoare, asigurând astfel siguranța și fiabilitatea structurilor proiectate. Performanța reprezintă, de asemenea, un aspect crucial, aplicația trebuind să fie rapidă și să răspundă prompt la interacțiunile utilizatorului, oferind o experiență de utilizare fluentă și eficientă. Nu în ultimul rând, mentenabilitatea este un obiectiv important, codul sursă al aplicației urmând să fie organizat și documentat într-un mod clar și concis, pentru a facilita modificările și actualizările viitoare și pentru a reduce costurile de întreținere.

### Metode de dezvoltare

Procesul de dezvoltare a aplicației va fi organizat și gestionat folosind o metodologie Agile, care pune accent pe dezvoltarea iterativă și incrementală. Această abordare presupune împărțirea proiectului în sprinturi de două săptămâni, în care echipa va planifica și executa un set specific de sarcini, livrând funcționalități într-un mod incremental. Comunicarea constantă în cadrul echipei și feedback-ul continuu din partea utilizatorilor și a altor părți interesate vor fi elemente esențiale ale procesului de dezvoltare, asigurând că aplicația evoluează în direcția corectă și satisface nevoile utilizatorilor.

### Strategii de arhitectură

Arhitectura aplicației va fi definită de o serie de strategii cheie, menite să asigure scalabilitatea, flexibilitatea și robustețea sistemului. Arhitectura stratificată reprezintă o strategie fundamentală, aplicația urmând să fie împărțită în straturi distincte (prezentare, logică de afaceri, acces la date), fiecare strat având o responsabilitate specifică. Această separare a preocupărilor facilitează mentenabilitatea, permițând modificarea unui strat fără a afecta semnificativ celelalte. Utilizarea de framework-uri și biblioteci existente este o altă strategie importantă, aplicația urmând să se bazeze pe componente software testate și validate (cum ar fi PyQt pentru interfața grafică și SQLite pentru baza de date), accelerând astfel procesul de dezvoltare și reducând riscul de erori. Nu în ultimul rând, modularizarea reprezintă o strategie esențială, aplicația urmând să fie împărțită în module independente, fiecare modul implementând o funcționalitate specifică. Această abordare facilitează dezvoltarea paralelă a diferitelor părți ale aplicației, reutilizarea codului și mentenabilitatea sistemului.

## Arhitectura Sistemului și Proiectarea Arhitecturii

### Arhitectură hardware

Aplicația este proiectată pentru a fi o aplicație desktop, ceea ce înseamnă că va rula pe un singur computer. Arhitectura hardware este, prin urmare, centralizată, cu toate componentele aplicației (interfață, logică, bază de date) executându-se pe același sistem. Configurația hardware minimă recomandată include un procesor Intel i5, 8GB RAM și 256GB spațiu de stocare pentru a asigura o performanță adecvată.

### Arhitectură software

Arhitectura software a aplicației este compusă din mai multe componente. Interfața cu utilizatorul este dezvoltată folosind biblioteca PyQt6, care permite crearea de interfețe grafice bogate și interactive. Logica de calcul este implementată în limbajul de programare Python, ales pentru claritatea și flexibilitatea sa. Datele sunt stocate și gestionate folosind SQLite, un sistem de baze de date relaționale încorporat, care nu necesită un server separat și simplifică implementarea. Alte biblioteci Python pot fi utilizate pentru funcționalități specifice, cum ar fi generarea de rapoarte PDF.

### Arhitectura informațiilor

Informațiile gestionate de aplicație includ datele de intrare furnizate de utilizator (de exemplu, dimensiunile elementelor structurale, proprietățile materialelor), datele rezultate din calculele structurale (de exemplu, forțe, momente, armături) și metadatele asociate proiectelor (de exemplu, numele proiectului, data creării). Aceste date sunt stocate local într-o bază de date SQLite.

### Arhitectura de comunicații interne

Deoarece aplicația este proiectată să funcționeze ca o aplicație desktop independentă, comunicarea internă între componente se realizează prin mecanismele oferite de sistemul de operare și de limbajul de programare Python. Nu sunt necesare protocoale de comunicare în rețea specifice.

### Diagrama de arhitectură a sistemului

Instrucțiuni: Utilizând proiectarea hardware, software, de comunicații și de informații descrise mai sus, redați structura generală și integrată a sistemului.

## Proiectarea sistemului

### Proiectarea bazei de date

Proiectarea bazei de date reprezintă un element crucial al arhitecturii aplicației, având în vedere rolul său central în stocarea tuturor datelor necesare pentru funcționarea sistemului. Pentru implementarea bazei de date, se va utiliza un sistem de gestionare a bazelor de date relaționale, și anume SQLite, datorită avantajelor sale în ceea ce privește simplitatea și eficiența. Baza de date va fi organizată într-o structură de tabele, unde fiecare tabel va reprezenta o entitate specifică din cadrul aplicației. Exemple de entități includ elementele structurale analizate, materialele utilizate în construcții și utilizatorii aplicației. Fiecare tabel va fi compus din coloane, iar fiecare coloană va defini un atribut specific al entității reprezentate de tabel. De exemplu, în tabelul care reprezintă elementele structurale, coloanele vor stoca dimensiunile elementului, tipul acestuia și alte proprietăți relevante. Pentru a asigura integritatea și consistența datelor, se vor defini relații între tabele, stabilind modul în care entitățile sunt conectate. De exemplu, se va stabili o relație între tabelul utilizatorilor și tabelul proiectelor, indicând ce utilizator a creat fiecare proiect. În plus, se va crea un dicționar de date, care va servi drept documentație centralizată pentru toate elementele de date utilizate în baza de date, specificând numele, tipul, lungimea și alte caracteristici importante ale fiecărui element.

#### Obiecte de date și structuri de date rezultante

În cadrul aplicației, obiectele de date principale manipulate includ elemente structurale precum grinzi și stâlpi, materiale de construcție precum beton și oțel, informații despre utilizatori și datele specifice ale proiectelor. Pentru a stoca eficient aceste obiecte de date, s-au utilizat structuri de date adecvate. Tabelele au fost utilizate pentru a organiza datele despre entități, de exemplu, un tabel "elements" pentru a stoca informații despre elementele structurale și un tabel "materials" pentru a stoca proprietățile materialelor. Atributele fiecărei entități au fost stocate folosind tipuri de date specifice, cum ar fi numere reale pentru dimensiuni, șiruri de caractere pentru nume și date calendaristice pentru datele de creare. Relațiile dintre tabele au fost definite pentru a reprezenta asocierile logice dintre entități, de exemplu, relația dintre un utilizator și proiectele pe care le-a creat. Structurile de date rezultante au fost adaptate la cerințele specifice ale fiecărui obiect de date. De exemplu, tabelul "elements" a fost proiectat să includă coloane pentru a stoca tipul elementului, dimensiunile sale, materialele utilizate și rezultatele calculelor structurale relevante.

#### Fișiere și baze de date

Modelul de date fizic a reprezentat implementarea structurilor de date în SQLite. Acest lucru a implicat crearea fișierului bazei de date SQLite local și definirea tabelelor, a coloanelor acestora, a tipurilor de date utilizate pentru fiecare coloană și a constrângerilor de integritate, cum ar fi cheile primare, cheile străine și constrângerile de unicitate și non-nulitate. Pentru a optimiza performanța accesului la date, au fost specificați indici pentru anumite coloane.

##### Baze de date

Proiectarea detaliată a bazei de date SQLite a implicat definirea instrucțiunilor SQL CREATE TABLE pentru fiecare tabel necesar. Pentru fiecare coloană din tabele, a fost specificat tipul de date SQLite corespunzător, asigurând astfel stocarea adecvată a diferitelor tipuri de informații. Constrângerile de integritate, inclusiv “PRIMARY KEY”, “FOREIGN KEY”, “UNIQUE” și “NOT NULL”, au fost definite pentru a menține consistența și validitatea datelor stocate în baza de date. În anumite cazuri, au fost definite și indexuri folosind instrucțiuni “CREATE INDEX” pentru a accelera căutările și sortările de date.

##### Fișiere non-DBMS

Gestionarea datelor în aplicație s-a realizat în principal prin intermediul bazei de date SQLite. Fișierele non-DBMS au fost utilizate în principal în scopuri specifice. Unul dintre cele mai importante utilizări ale fișierelor non-DBMS a fost generarea de rapoarte, în special fișiere PDF care conțin breviarele de calcul. În anumite cazuri, fișierele non-DBMS au fost utilizate și pentru stocarea temporară a datelor sau a configurațiilor specifice ale utilizatorului. Pentru fiecare tip de fișier non-DBMS utilizat, au fost specificate caracteristici precum numele și locația fișierului, scopul său (intrare, ieșire sau temporar), modulul software care a fost responsabil pentru citirea și scrierea fișierului și structura internă a fișierului, inclusiv formatul datelor, delimitatorii și tipurile de date utilizate.

### Conversii de date

Conversiile de date au reprezentat o necesitate în anumite etape ale funcționării aplicației. Cele mai frecvente tipuri de conversii de date au inclus importul și exportul datelor din și în alte formate, cum ar fi CSV sau Excel, pentru a facilita schimbul de informații cu alte aplicații. De asemenea, au fost implementate conversii între diferite unități de măsură, de exemplu, conversia între metri și milimetri, pentru a asigura flexibilitatea utilizării aplicației în diferite contexte. Nu în ultimul rând, au fost efectuate conversii de formatare a datelor pentru a le pregăti pentru afișare în interfața cu utilizatorul sau pentru includerea în rapoartele generate de aplicație. Pentru fiecare tip de conversie, au fost specificate detaliile relevante, cum ar fi datele de intrare și formatul lor, datele de ieșire și formatul lor, algoritmul sau metoda de conversie utilizată și, opțional, rutinele software specifice responsabile pentru efectuarea conversiei.

### Interfețe utilizator

Interfețele utilizator ale aplicației au fost proiectate ținând cont de diferitele clase de utilizatori care vor interacționa cu sistemul. Aceste clase de utilizatori au inclus în principal studenții, inginerii și, opțional, profesorii. Fiecare clasă de utilizatori a fost caracterizată prin roluri și responsabilități specifice, niveluri diferite de expertiză în domeniul calculului structural și fluxuri de lucru tipice distincte. Pentru fiecare clasă de utilizatori, au fost definite cerințele specifice de interfață, asigurând astfel că aplicația este adaptată nevoilor și preferințelor fiecărui grup. De asemenea, au fost furnizate estimări ale numărului total de utilizatori anticipați și ale numărului maxim de utilizatori simultani, informații esențiale pentru dimensionarea corectă a resurselor sistemului și asigurarea performanței optime.

#### Intrări

Utilizatorii au introdus date în aplicație prin intermediul diferitelor mecanisme. Cea mai frecventă metodă de introducere a datelor a fost utilizarea formularelor din interfața grafică (GUI), care au permis specificarea proprietăților elementelor structurale, a caracteristicilor materialelor, a valorilor încărcărilor și a altor parametri relevanți pentru calculele structurale. În anumite cazuri, a fost oferită și posibilitatea de a importa date din fișiere externe, facilitând astfel transferul de informații din alte aplicații sau surse de date. Pentru fiecare tip de intrare, au fost specificate detaliile relevante, cum ar fi ecranele sau formularele GUI utilizate (cu aspectul și elementele lor componente), elementele de date individuale (cu tipul, lungimea, formatul și regulile de validare asociate) și controalele de introducere a datelor implementate pentru a asigura validitatea și corectitudinea datelor introduse, precum și mesajele de eroare afișate pentru a ghida utilizatorii în cazul introducerii de date incorecte.

#### Ieșiri

Aplicația a generat diferite tipuri de ieșiri pentru a prezenta rezultatele calculelor și a furniza informații utile utilizatorilor. Cea mai frecventă formă de ieșire a fost afișarea rezultatelor direct în interfața grafică, utilizând tabele, grafice și alte elemente vizuale pentru a facilita interpretarea și înțelegerea datelor. De asemenea, aplicația a generat rapoarte detaliate în format PDF, denumite breviare de calcul, care au inclus toate datele de intrare, rezultatele calculelor și alte informații relevante pentru documentarea procesului de proiectare. În anumite cazuri, a fost oferită și posibilitatea de a exporta datele în fișiere externe, permițând utilizatorilor să le utilizeze în alte aplicații sau să le arhiveze. Pentru fiecare tip de ieșire, au fost specificate caracteristicile relevante, cum ar fi formatul și structura ieșirii, conținutul detaliat (elementele de date incluse), scopul ieșirii și utilizatorii țintă, precum și eventualele restricții de acces sau considerații de securitate aplicabile.

### Proiectarea interfețelor cu utilizatorul

Proiectarea interfețelor cu utilizatorul a fost ghidată de o serie de principii fundamentale, menite să asigure o experiență de utilizare optimă. Intuitivitatea a reprezentat un principiu cheie, interfața fiind proiectată pentru a fi ușor de învățat și de utilizat, chiar și de către utilizatorii cu experiență limitată în aplicații de calcul structural. Eficiența a fost un alt principiu important, interfața fiind optimizată pentru a permite utilizatorilor să efectueze sarcinile dorite rapid și cu minimum de efort. Feedback-ul prompt și clar a fost considerat esențial, aplicația furnizând informații utile utilizatorilor despre starea sistemului și rezultatele acțiunilor lor. Consistența a fost menținută pe întregul parcurs al aplicației, asigurându-se că elementele de interfață au un aspect și un comportament similar în diferite contexte. Accesibilitatea a fost luată în considerare în măsura posibilităților unei aplicații desktop, respectându-se standardele și recomandările relevante. În implementarea interfeței, au fost utilizate elemente GUI standard, cum ar fi ferestre, meniuri, butoane, formulare, tabele și grafice, asigurându-se astfel familiaritatea utilizatorilor cu aplicația. Au fost specificate cu atenție aspecte precum layout-ul ecranelor, culorile utilizate, fonturile și alte detalii de design, pentru a crea o interfață atractivă și funcțională.

## Scenarii de utilizare

Această secțiune descrie interacțiunile tipice dintre utilizatori și aplicație, ilustrând modul în care aceștia vor utiliza diversele funcționalități pentru a îndeplini sarcinile dorite. Scenariile de utilizare prezintă un flux operațional al sistemului, oferind descrieri pas cu pas ale secvențelor de acțiuni și răspunsuri ale sistemului în diferite contexte. Aceste scenarii au rolul de a lega toate părțile componente ale aplicației, de a defini interacțiunile dintre utilizatori și sistem și de a clarifica comportamentul așteptat al aplicației în diverse situații.

**6.1 Autentificare utilizator**

Procesul de autentificare a utilizatorului reprezintă punctul de intrare în aplicație și asigură accesul securizat la funcționalitățile acesteia. Atunci când un utilizator deschide aplicația, sistemul afișează ecranul de autentificare, solicitând utilizatorului să introducă credențialele sale. Aceste credențiale constau, de obicei, într-un nume de utilizator și o parolă, care sunt apoi validate de aplicație prin compararea lor cu cele stocate în baza de date. În cazul în care credențialele introduse de utilizator sunt valide, sistemul autentifică utilizatorul și afișează ecranul principal al aplicației, oferind acces la funcționalitățile disponibile. Dacă credențialele sunt invalide, sistemul afișează un mesaj de eroare clar și informativ, solicitând utilizatorului să reintroducă datele de autentificare corecte.

**6.2 Gestionarea proiectelor**

Aplicația oferă funcționalități robuste pentru gestionarea proiectelor, permițând utilizatorilor să organizeze și să stocheze eficient calculele structurale. Pentru a crea un proiect nou, utilizatorul va selecta opțiunea corespunzătoare din meniul aplicației. Sistemul va solicita apoi utilizatorului să introducă un nume unic pentru proiect, permițându-i să identifice și să organizeze proiectele în mod adecvat. După ce utilizatorul a introdus numele proiectului, aplicația va crea un nou proiect în baza de date și îl va deschide, punându-l la dispoziția utilizatorului pentru a introduce datele și a efectua calcule.

**6.3 Efectuarea unui calcul structural**

Aplicația permite utilizatorilor să efectueze o varietate de calcule structurale pentru a analiza și proiecta elemente de construcție. Pentru a iniția un calcul, utilizatorul va selecta tipul specific de calcul dorit din lista opțiunilor disponibile, cum ar fi calculul forței tăietoare, al momentului încovoietor sau al armăturii necesare. Apoi, utilizatorul va selecta elementul structural pentru care se va efectua calculul, specificând tipul acestuia (de exemplu, grindă, stâlp) și caracteristicile sale geometrice. Ulterior, utilizatorul va introduce proprietățile relevante ale materialelor utilizate în elementul structural, cum ar fi clasele de beton și oțel. În funcție de tipul de calcul selectat, utilizatorul va introduce și alți parametri necesari, cum ar fi încărcările aplicate elementului sau coeficienții de siguranță. După ce toate datele de intrare au fost furnizate, utilizatorul va iniția calculul, iar aplicația va efectua calculele necesare pe baza normativelor și formulelor relevante. Rezultatul calculului va fi apoi afișat utilizatorului într-un format clar și ușor de interpretat, permițându-i să evalueze comportamentul structural al elementului.

**6.4 Generarea unui raport PDF**

Pentru a facilita documentarea și partajarea rezultatelor calculelor structurale, aplicația oferă funcționalitatea de generare a rapoartelor în format PDF. După ce utilizatorul a efectuat un calcul, acesta poate selecta opțiunea de a genera un raport, iar aplicația va crea automat un fișier PDF care va conține toate datele de intrare utilizate în calcul, precum și rezultatul obținut. Înainte de a salva raportul, aplicația poate oferi utilizatorului posibilitatea de a previzualiza documentul, asigurându-se că conținutul și formatarea sunt conforme cu așteptările sale. Utilizatorul poate apoi salva fișierul PDF într-o locație aleasă de acesta, permițându-i să îl arhiveze, să îl tipărească sau să îl distribuie altor părți interesate.

**6.5 Editarea profilului utilizatorului**

Aplicația permite utilizatorilor să își gestioneze informațiile personale prin intermediul funcționalității de editare a profilului. Utilizatorul poate accesa ecranul de profil, unde aplicația va afișa informațiile curente ale profilului, cum ar fi numele, adresa de email și alte detalii relevante. Utilizatorul poate apoi modifica orice informație care necesită actualizare. După ce utilizatorul a efectuat modificările dorite, acesta poate salva aceste modificări, iar aplicația va actualiza în consecință informațiile profilului utilizatorului în baza de date.

## Proiectare de detaliu

### Proiectare hardware de detaliu

Având în vedere că aplicația este proiectată pentru a funcționa ca o aplicație desktop, cerințele hardware detaliate se concentrează pe specificațiile computerului pe care va fi instalată. Se recomandă ca utilizatorii să dispună de un PC sau laptop cu un procesor Intel Core i5 de a 10-a generație sau un procesor echivalent, 8GB de memorie RAM DDR4 și un dispozitiv de stocare SSD cu o capacitate de 256GB. Pentru a asigura o afișare adecvată a interfeței grafice, se recomandă un monitor cu o rezoluție minimă de 1920x1080 pixeli. Aceste specificații hardware minime sunt considerate necesare pentru a obține o performanță acceptabilă a aplicației și o experiență de utilizare fluentă

### Proiectare software de detaliu

Arhitectura software a aplicației este compusă din mai multe module distincte, fiecare având o responsabilitate specifică în cadrul sistemului. Modulul de interfață cu utilizatorul (UI) este responsabil pentru afișarea informațiilor către utilizator și pentru gestionarea interacțiunilor utilizatorului cu aplicația. Acest modul va fi dezvoltat folosind biblioteca PyQt6 și limbajul de programare Python, utilizând componente precum ferestre, formulare, butoane și grafice pentru a prezenta rezultatele calculelor într-un mod vizual și intuitiv.

Modulul de calcul structural reprezintă nucleul aplicației și este responsabil pentru efectuarea calculelor specifice domeniului, cum ar fi calculul forței tăietoare, al momentului încovoietor și al altor parametri structurali relevanți. Acest modul va fi implementat în Python, utilizând biblioteci științifice precum NumPy și SciPy pentru a efectua eficient calculele numerice. Modulul de gestionare a datelor este responsabil pentru stocarea și accesarea datelor persistente ale aplicației, incluzând informații despre proiecte, materiale și utilizatori. Pentru a implementa această funcționalitate, se va utiliza SQLite, un sistem de gestionare a bazelor de date relaționale încorporat, iar interacțiunile cu baza de date vor fi realizate prin intermediul unor clase Python specifice.

Modulul de generare a rapoartelor va fi responsabil pentru crearea de rapoarte detaliate ale calculelor în format PDF, utilizând biblioteca ReportLab și limbajul Python pentru a formata datele și a genera documentele PDF. Pentru fiecare modul și componentă software specifică, se vor specifica detaliile relevante pentru implementare, cum ar fi clasele și funcțiile principale, algoritmii utilizați (în special în modulul de calcul structural), structurile de date interne utilizate, mecanismele de gestionare a erorilor și excepțiilor și interacțiunile cu alte module ale aplicației.

### Proiectare detaliată de securitate

Securitatea aplicației este un aspect important, în special în ceea ce privește protecția datelor sensibile ale utilizatorilor. Pentru a asigura securitatea procesului de autentificare, se va utiliza o metodă de stocare sigură a parolelor, cum ar fi utilizarea de funcții hash (de exemplu, SHA-256) pentru a stoca hash-urile parolelor în loc de parolele în clar. În plus, se poate implementa un mecanism de blocare a contului după un număr specificat de încercări de autentificare eșuate, pentru a preveni atacurile de forță brută. Autorizarea, adică controlul accesului la datele și funcționalitățile aplicației, va fi implementată pentru a asigura că utilizatorii au acces doar la informațiile și operațiunile pentru care sunt autorizați. De exemplu, se va implementa un control al accesului la datele proiectelor, astfel încât fiecare utilizator să aibă acces doar la proiectele pe care le-a creat sau la proiectele la care are permisiunea de a accesa. În plus, se poate lua în considerare implementarea criptării fișierului bazei de date pentru a proteja datele stocate local de accesul neautorizat.

### Proiectare de detaliu pentru performanța sistemului

Performanța reprezintă un aspect crucial al aplicației, deoarece influențează direct experiența utilizatorului. Pentru a asigura o performanță optimă, se vor implementa diverse strategii de proiectare. În primul rând, se va acorda o atenție deosebită optimizării calculelor structurale, utilizând algoritmi eficienți și structuri de date adecvate pentru a reduce timpul de calcul. În anumite cazuri, se poate lua în considerare utilizarea de tehnici de calcul paralel, dacă este fezabil și adecvat pentru tipul de calcul efectuat. În plus, se vor optimiza accesul la datele stocate în baza de date, utilizând indexarea tabelelor și alte tehnici de optimizare a interogărilor pentru a reduce timpul necesar pentru a accesa și a prelua datele. De asemenea, se va acorda atenție utilizării eficiente a memoriei, evitând alocarea excesivă de memorie și eliberând memoria neutilizată pentru a minimiza impactul asupra performanței sistemului.

### Proiectare detaliată a comunicațiilor interne (între componente)

Deoarece aplicația este proiectată ca o aplicație desktop independentă, comunicarea internă între componentele sale se va realiza în principal prin intermediul mecanismelor oferite de limbajul de programare Python și de sistemul de operare. Modulele aplicației vor comunica între ele prin apeluri directe de funcții, transmițând datele necesare sub formă de structuri de date definite. Nu vor fi necesare protocoale de comunicare în rețea specifice pentru comunicarea internă a aplicației.

## Controale pentru verificarea integrității sistemului

Pentru a asigura fiabilitatea și corectitudinea aplicației, se vor implementa o serie de controale pentru verificarea integrității sistemului și a datelor. Acestea vor include validarea datelor de intrare, care va preveni introducerea de date incorecte sau invalide de către utilizatori prin verificări ale tipului, formatului, intervalului și lungimii datelor. De asemenea, se vor utiliza constrângeri de integritate a bazei de date, cum ar fi chei primare, chei străine, constrângeri de unicitate și constrângeri NOT NULL, pentru a asigura consistența și validitatea datelor stocate. Jurnalizarea și auditarea reprezintă un alt mecanism important de control, aplicația urmând să țină un jurnal al evenimentelor relevante, cum ar fi autentificările utilizatorilor, modificările datelor și erorile, permițând astfel auditarea activității utilizatorilor și identificarea eventualelor probleme. Nu în ultimul rând, se vor implementa mecanisme de gestionare a erorilor și excepțiilor, pentru a preveni blocarea aplicației și pentru a afișa mesaje de eroare informative pentru utilizatori, facilitând astfel diagnosticarea și rezolvarea problemelor.

Anexa A: Gestiunea modificărilor documentului

Instrucțiuni: Furnizați informații despre modul în care dezvoltarea și distribuția documentului va fi controlată și urmărită. Utilizați tabelul de mai jos pentru a furniza numărul de versiune, data versiunii, autorul/deținătorul versiunii și o scurtă descriere a motivului pentru crearea versiunii revizuite.

Tabel 1 – Înregistrarea modificărilor asupreaa documentului curent

| versiune | Data | Autorul/Deținătorul | Descriere |
| --- | --- | --- | --- |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |
| <X.X> | <ZZ/LL/AAAA> | <nume autor> | <Descrierea modificării> |

Anexa B: Acronime

*Instrucțiuni: Furnizați o listă de acronime și traduceri literale asociate utilizate în cadrul documentului. Enumerați acronimele în ordine alfabetică folosind un format tabular, așa cum este ilustrat mai jos.*

Tabel 2 - Acronime

| Acronim | Forma completă |
| --- | --- |
| <Acronim> | <Forma completă> |
| <Acronim> | <Forma completă> |
| <Acronim> | <Forma completă> |

Anexa C Documente la care se face referire

*Instrucțiuni: Sintetizați relația acestui document cu alte documente relevante. Furnizați informații de identificare pentru toate documentele folosite pentru a ajunge la și/sau referite în acest document (de exemplu, documente conexe și/sau asociate, documente prealabile, documentație tehnică relevantă, etc.).*

Tabel 3 – Documente la care se facce referire

| Nume document | Locație sau URL | Dată emitere document |
| --- | --- | --- |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |
| < Nume document > | <Locație sau URL> | <ZZ/LL/AAAA> |