Sistem inteligent de monitorizare și ajustare a parametrilor de confort într-un spațiu de birou

# Documentul de proiectare

[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)

1. Introducere

Documentul de Proiectare a Sistemului **„Sistem de Monitorizare și Ajustare Inteligentă a Parametrilor din Spații de Birou”** descrie modul în care cerințele funcționale și non-funcționale identificate în faza de analiză se transformă în specificații de proiectare detaliată a aplicației software.

Scopul acestui sistem este de a permite monitorizarea în timp real a unor parametri de mediu (temperatură, umiditate, luminozitate, zgomot, calitate aer) cu ajutorul senzorilor fizici, dar și ajustarea valorilor afișate în interfață în funcție de voturile subiective ale utilizatorilor. Astfel, se creează o punte între datele obiective măsurate și percepția utilizatorilor asupra confortului.

Sistemul este dezvoltat sub forma unei aplicații desktop în Python (Tkinter), cu interfață intuitivă, conectată la o bază de date SQLite în care se salvează automat toate interacțiunile: voturi, comentarii și istoric. Logica de reglare automată se bazează pe un algoritm care modifică valorile simulate doar după minim 3 voturi consecutive în aceeași direcție, independent de valorile reale.

Acest document detaliază arhitectura sistemului la un nivel înalt, descriind componentele hardware și software, interacțiunea utilizatorului cu aplicația, fluxurile de date și scenariile de funcționare. Este inclusă și o viziune asupra posibilelor extensii viitoare (ex: integrare cu actuatoare reale – lumini, ventilație).

**Securitatea** este tratată printr-un sistem de autentificare cu parolă criptată și separarea rolurilor utilizatorilor. Nu sunt colectate date personale sensibile. Aplicația este destinată în primul rând uzului educațional sau experimental într-un birou inteligent.

1.1 Scopul documentului

Scopul acestui document este de a defini și detalia arhitectura aplicației de monitorizare și ajustare automată a parametrilor de mediu într-un birou, precum și logica de funcționare a fiecărei componente software și a interacțiunilor dintre utilizatori, senzori și interfață.

Documentul urmărește să ofere o imagine clară asupra modului în care sistemul combină datele reale cu preferințele subiective ale utilizatorilor, pentru a crea un mediu adaptat confortului colectiv. Se documentează, de asemenea, modul de gestionare a voturilor, salvarea datelor și istoricul feedback-ului.

Aplicația este dezvoltată incremental, cu posibilitatea de extindere viitoare spre controlul real al mediului (prin actuatoare). Informațiile din acest document sunt utile pentru dezvoltatori, profesori, evaluatori sau alți contributori care doresc să înțeleagă logica și structura sistemului.

2. Prezentare generală și abordări de proiectare

Această secțiune descrie principiile și strategiile aplicate în proiectarea sistemului inteligent de monitorizare și ajustare a parametrilor dintr-un spațiu de birou, prin combinarea datelor reale (senzori) cu preferințele subiective (voturi).

2.1 Prezentare generală

Aplicația urmărește dezvoltarea unui sistem hibrid, care îmbină:

* măsurători obiective provenite de la senzori (temperatură, umiditate, luminozitate, zgomot, calitate aer),
* cu un mecanism de ajustare bazat pe voturi colective, ce reflectă confortul perceput al utilizatorilor.

Arhitectura generală a sistemului este compusă din:

* o interfață grafică (GUI) intuitivă realizată cu Tkinter;
* o bază de date SQLite locală pentru salvarea feedback-ului și a conturilor;
* un set de reguli logice care declanșează modificări simulate asupra valorilor afișate pe ecran, doar după minim 3 voturi consecutive în aceeași direcție.

Abordarea de proiectare este modulară:

* **Interfața LoginWindow**: autentificare și creare cont;
* **Clasa BirouInteligent**: gestionează UI-ul principal și voturile;
* **Clasa BazaDeDate**: persistă datele și oferă interogări rapide.

Această compartimentare permite o dezvoltare incrementală, precum și posibile extensii viitoare precum integrarea cu microcontrolere, actuatoare fizice sau predicții pe baza istoricelor.

**2.2 Presupuneri / Constrângeri / Riscuri**

**2.2.1 Presupuneri**

Pentru realizarea și rularea corectă a aplicației, au fost formulate următoarele presupuneri:

✅ Aplicația va rula local, pe un sistem compatibil cu Python 3.x și Tkinter (Windows/Linux).

✅ Senzorii reali vor fi integrați doar pentru citire (fără feedback fizic înapoi spre mediu).

✅ Utilizatorii sistemului sunt familiarizați cu o interfață simplă cu butoane și scale (slidere).

✅ Se presupune existența unui singur tip de utilizator standard (cu user și parolă).

✅ Nu se colectează date personale sensibile; sistemul funcționează anonimizat.

**2.2.2 Constrângeri**

Constrângerile proiectului au influențat alegerile tehnologice și funcționale:

🔹 **Mediu hardware/software**  
→ Sistemul trebuie să funcționeze pe PC-uri obișnuite, cu consum redus de resurse (RAM, CPU). Nu se bazează pe GPU sau cloud.

🔹 **Mediu utilizator final**  
→ Aplicația este destinată unui mediu educațional sau de birou cu acces local. Nu se cere experiență tehnică din partea utilizatorului.

🔹 **Disponibilitatea resurselor**  
→ Senzorii utilizați (reali sau simulați) trebuie să fie disponibili fizic sau înlocuiți temporar prin valori generate random.

🔹 **Conformitatea cu standardele**  
→ Nu se integrează protocoale industriale avansate (ex: MQTT, Modbus), ci doar structuri simple Python-SQLite.

🔹 **Interoperabilitate**  
→ Nu este planificată conectarea directă cu sisteme externe sau aplicații mobile.

🔹 **Cerințe de interfață/protocol**  
→ Interfața este locală, realizată în Tkinter, fără comunicații externe sau API-uri.

🔹 **Licențiere și distribuție**  
→ Proiectul este destinat uzului academic. Poate fi distribuit open-source fără restricții.

🔹 **Depozitare și distribuție date**  
→ Datele sunt stocate local într-un fișier .db. Nu există backup în cloud sau sincronizare.

**3. Considerații de proiectare**

**3.1 Obiective și linii directoare (ghiduri)**

Scopul principal al aplicației este crearea unui sistem inteligent de monitorizare a parametrilor unui spațiu de birouri, care să integreze atât date reale (de la senzori fizici) cât și opinii subiective (prin voturi de la utilizatori), în vederea ajustării valorilor afișate. Aplicația urmărește confortul colectiv și personalizarea mediului de lucru.

Obiective cheie:

* ✅ Simplitatea utilizării (interfață clară, intuitivă).
* ✅ Separarea logică între datele reale (senzori) și cele percepute (simulare ajustabilă).
* ✅ Scalabilitate și modularitate pentru integrarea ulterioară a sistemelor automate (ventilare, iluminare).
* ✅ Transparență și control: istoricul voturilor este accesibil fiecărui utilizator.

Liniile directoare includ:

* Toate ajustările se fac printr-un algoritm democratic: 3 voturi consecutive în aceeași direcție → declanșează modificarea.
* Baza de date trebuie să fie accesibilă rapid și eficient pentru citire/scriere (SQLite).
* Feedbackul trebuie să fie salvat în timp real, împreună cu utilizatorul, ora și valoarea percepută.

**3.2 Metode de dezvoltare**

Sistemul a fost proiectat iterativ și incremental, folosind metode orientate pe componente și interfețe. Limbajul de programare utilizat este Python, iar interfața este construită cu Tkinter.

Etapele de dezvoltare:

* 🔹 Prototipare rapidă a interfeței pentru testarea logicii de vot și vizualizare.
* 🔹 Integrare progresivă a componentelor: autentificare, salvare feedback, istoric.
* 🔹 Refactorizare modulară a clasei BirouInteligent pentru a permite extinderea cu noi parametri (ex: CO₂, VOC).

Contingențe:

* Interfața poate fi adaptată ușor pentru alte UI libraries (ex: PyQt, Kivy).
* Sistemul este pregătit pentru a integra senzori reali prin serial sau API.

**3.3 Strategii de arhitectură**

Deciziile majore de arhitectură:

* ✅ Separarea în 3 componente majore: interfață (LoginWindow), logica principală (BirouInteligent), stocare (BazaDeDate).
* ✅ Toate modificările asupra parametrilor se fac doar în interiorul clasei BirouInteligent → control centralizat.
* ✅ Algoritmul de voturi e configurabil (poți seta pragul de voturi sau coeficienții de ajustare).
* ✅ Salvarea feedbackului și accesarea utilizatorilor sunt complet abstractizate în clasa BazaDeDate.

Alte decizii semnificative:

* ✅ Alegerea SQLite: simplu, local, potrivit pentru prototipuri și testare fără server dedicat.
* ✅ Evitarea interfeței web pentru a păstra proiectul în zona aplicațiilor desktop didactice.
* ✅ Cod Python ușor de citit, comentat și adaptat de alți studenți în scop educațional.

**4. Arhitectura Sistemului și Proiectarea Arhitecturii**

Sistemul inteligent de monitorizare și ajustare a parametrilor de birou este structurat modular, cu separarea clară între componentele responsabile de interfață, logică de control, procesare date și persistarea acestora. Arhitectura generală urmărește o dezvoltare extensibilă, scalabilă și ușor de întreținut, folosind paradigme moderne bazate pe arhitectură client-server, gestionare locală a datelor (SQLite) și o logică de ajustare automată bazată pe feedback colectiv.

**4.1 Vedere logică**

Sistemul este împărțit logic în următoarele module:

* **Interfață utilizator (UI):** permite interacțiunea cu utilizatorii (autentificare, trimitere feedback, vizualizare date).
* **Modul de control (BirouInteligent):** preia și interpretează voturile utilizatorilor și ajustează valorile simulate.
* **Bază de date:** persistă informațiile despre utilizatori și feedback-ul acordat.
* **Modul de istoric și analiză:** folosit pentru afișarea evoluției parametrilor.

Comunicarea între module este realizată prin apeluri funcționale directe și scrieri în baza de date.

**4.2 Arhitectură hardware**

Deoarece sistemul rulează în întregime pe o aplicație desktop locală (Tkinter cu SQLite), arhitectura hardware este minimală:

* **Procesor și RAM:** un sistem mediu este suficient (ex: Intel i5, 8GB RAM).
* **Stocare locală:** minim 100MB pentru salvarea datelor SQLite.
* **Integrare viitoare:** posibilă extensie cu senzori reali prin porturi USB sau rețea (ex: DHT11, MQ135, etc.).

**4.3 Arhitectură software**

Sistemul este dezvoltat în Python, folosind următoarele componente:

* **Tkinter** – interfață grafică.
* **SQLite** – gestionare date persistente.
* **Modul personalizat (BirouInteligent):** cu funcții precum update\_temp\_feedback(), salveaza\_feedback() etc.
* **Modul de login:** autentificare și creare cont cu hash\_password().

Software-ul este organizat pe clase (ex: LoginWindow, BirouInteligent) și folosește apeluri clare pentru gestionarea feedback-ului, autentificare și salvare date. Structura este orientată pe MVC (Model-View-Controller), chiar dacă la nivel simplificat.

**4.4 Arhitectura informațiilor**

Datele stocate includ:

* **Date de utilizator:** username și parolă (criptată).
* **Feedback votat:** valori între -3 și +3 pentru fiecare parametru.
* **Istoric voturi:** pentru analiză ulterioară.
* **Parametri ajustați automat:** valori simulate pentru temperatură, umiditate, lumină, zgomot, aer.

Nu se stochează date cu caracter personal sensibil (ex: nume real, email, etc.).

**4.5 Arhitectura de comunicații interne**

Aplicația este locală și nu necesită conexiuni în rețea. Totuși, comunicația internă între componente se realizează astfel:

* **Apeluri directe între clase și funcții.**
* **Persistarea datelor în baza de date SQLite.**

Într-o extensie viitoare, comunicarea cu senzori reali sau servere web externe se poate realiza prin socket TCP/IP sau REST API.

**4.6 Diagrama de arhitectură a sistemului**

Diagrama finală va include toate modulele și relațiile dintre ele:

* **UI → BirouInteligent → BazaDeDate**
* **LoginWindow → BirouInteligent → Feedback**
* **BirouInteligent ↔ Istoric voturi**

Această diagramă reflectă clar arhitectura modulară, fluxul de date și independența componentelor principale.

A diagram of a diagram

AI-generated content may be incorrect.

**5. Proiectarea sistemului**

**5.1 Proiectarea bazei de date**

Sistemul utilizează o bază de date SQLite pentru stocarea conturilor utilizatorilor, feedbackului trimis (pentru fiecare parametru – temperatură, lumină, umiditate, zgomot, calitate aer) și opțional istoricul ajustărilor automate. Structura este simplă, optimizată pentru aplicații de tip desktop standalone.

**5.1.1 Obiecte de date și structuri de date rezultate**

* users (id, username, password\_hash)
* feedback (id, user\_id, tip\_parametru, valoare, timestamp)
* istoric\_valori (id, tip\_parametru, valoare\_simulata, timestamp)

**5.1.2 Fișiere și baze de date**

**5.1.2.1 Baze de date**

Baza de date SQLite este stocată local. Se realizează backup manual sau la închiderea aplicației. Tabelele sunt optimizate pentru inserții frecvente și interogări după user\_id și tip\_parametru.

**5.1.2.2 Fișiere non-DBMS**

* Fișiere CSV pot fi generate pentru exportul feedbackului sau raportarea voturilor.
* Fiecare rând include: tip\_parametru, valoare, user\_id, data\_ora.
* Sunt utilizate temporar în scop de backup sau analiză offline.

**5.2 Conversii de date**

Datele introduse de utilizatori prin aplicație (slider) sunt convertite în format numeric (-3 la +3), apoi salvate ca valori întregi în baza de date. Acestea sunt ulterior traduse în ajustări procentuale pentru fiecare parametru. La export, datele sunt convertite în format CSV cu etichete semnificative.

**5.3 Interfețe utilizator**

Aplicația dispune de două tipuri de interfețe:

* **LoginWindow:** pentru autentificare și creare cont.
* **BirouInteligent:** interfața principală, cu slidere pentru feedback, valori simulate, comentarii și acces la istoric.

Sistemul este destinat unui singur utilizator simultan, rulând local pe sistemul desktop.

**5.3.1 Intrări**

* Slider pentru fiecare parametru cu valori între -3 și +3.
* Câmp pentru comentariu opțional.
* Buton „Trimite feedback”.
* Pagina „Istoric voturi” care permite interogarea valorilor anterioare.

**5.3.2 Ieșiri**

* Valorile ajustate pentru fiecare parametru se afișează în timp real.
* În funcție de medie, se aplică o ajustare procentuală asupra valorii simulate.
* Feedbackul este salvat și afișat în pagina de istoric.

**5.4 Proiectarea interfețelor cu utilizatorul**

* Interfață principală ergonomică, pe verticală, cu slidere colorate și etichete clare.
* Culorile sliderelor se schimbă în funcție de valoare (roșu, galben, verde).
* Notificări dinamice (ex: alertă dacă temperatura > 30°C).
* Comentariile sunt afișate în partea dreaptă, într-un panou scrollabil.
* Fereastra de login simplă, cu opțiuni de creare cont și verificare parolă.

**6. Scenarii de utilizare**

Scenariile de utilizare ilustrează modul în care sistemul este folosit de utilizatori pentru a interacționa cu interfața, a furniza feedback și a vizualiza starea parametrilor. Ele reflectă aspectele operaționale ale aplicației, urmărind fluxurile de date de la interacțiunea inițială până la salvarea și procesarea valorilor în sistem.

**Scenariul 1 – Autentificarea utilizatorului**

**Actor:** Utilizator  
**Descriere:** La deschiderea aplicației, utilizatorul este întâmpinat cu o fereastră de login. Introduce un nume de utilizator și o parolă. Parola este criptată și verificată în baza de date.  
**Rezultat așteptat:** Dacă autentificarea reușește, se deschide interfața principală. În caz contrar, utilizatorul primește un mesaj de eroare.

**Scenariul 2 – Acordarea de feedback pentru parametrii de confort**

**Actor:** Utilizator  
**Descriere:** Utilizatorul reglează sliderele pentru temperatură, umiditate, lumină, aer și zgomot în funcție de percepția personală. Poate adăuga și un comentariu textual. Apasă pe butonul „Trimite feedback”.  
**Rezultat așteptat:** Valorile și comentariul sunt salvate în baza de date. Sistemul verifică dacă este necesară ajustarea valorilor simulate pe baza feedbackului acumulat.

**Scenariul 3 – Vizualizarea valorilor simulate și istoricului de voturi**

**Actor:** Utilizator  
**Descriere:** Accesează secțiunea „Istoric voturi”, unde poate vedea evoluția voturilor proprii sau ale altor utilizatori (dacă e admin). Diagrama este actualizată în timp real.  
**Rezultat așteptat:** Utilizatorul poate analiza trenduri și corela reacțiile sale cu starea generală a mediului simulat.

**Scenariul 4 – Ajustarea valorii simulate pe baza unui algoritm de majoritate**

**Actor:** Sistem (automat)  
**Descriere:** La fiecare 3 voturi consistente (în aceeași direcție) pentru un anumit parametru, sistemul ajustează valoarea simulată cu un maxim de 10%.  
**Rezultat așteptat:** Valoarea vizibilă în interfață se modifică subtil, reflectând opinia colectivă.

**Scenariul 5 – Consultarea feedbackului general (doar admin)**

**Actor:** Administrator  
**Descriere:** Administratorul poate accesa secțiunea de feedback general din meniu, unde poate vizualiza toate voturile trimise, împreună cu statistici privind preferințele utilizatorilor.  
**Rezultat așteptat:** Adminul are o imagine clară asupra nivelului de confort raportat și poate lua decizii pentru adaptarea ulterioară a aplicației.

**7. Proiectarea sistemului**

**7.1 Proiectarea hardware de detaliu**

Aplicația propusă se bazează pe o arhitectură hardware simplificată, centrată pe un microcontroller (NodeMCU ESP8266/ESP32) ce preia date de la o serie de senzori pentru parametri de mediu (temperatură, umiditate, calitate aer, lumină, zgomot). Acești senzori sunt integrați în sistemul fizic de monitorizare și oferă date în timp real prin conexiune Wi-Fi.

**Componentele hardware utilizate**:

* Senzor DHT22 – măsurarea temperaturii și umidității.
* Senzor MQ135 – calitatea aerului (gaz metan, amoniac etc.).
* Senzor BH1750 – măsurare lumină ambientală.
* Senzor de zgomot (microfon analogic).
* Placă de dezvoltare NodeMCU ESP32 – comunicare Wi-Fi, prelucrare locală.
* Alimentare prin USB 5V (de la priză sau powerbank).

Comunicarea dintre senzori și modulul central se realizează prin GPIO-uri, utilizând protocoale I2C, analog, sau digital. Toate componentele sunt amplasate pe o placă de test (breadboard) și pot fi mutate pe un PCB pentru o versiune finală compactă.

**7.2 Proiectarea software de detaliu**

Sistemul software este compus dintr-un backend în Python (pentru logica de colectare, salvare și procesare date) și o interfață vizuală realizată în Tkinter.

**Servicii software principale**:

* Serviciu de login/creare cont (LoginWindow).
* Serviciu de colectare și trimitere feedback (update\_feedback).
* Algoritm de ajustare a valorilor simulate (verificare contor voturi, ajustare, salvare).
* Serviciu de salvare în SQLite și istoric feedback.

**Structură și logică**:

* Feedbackul utilizatorului este gestionat printr-o logică bazată pe voturi repetate.
* Valoarea este ajustată după 3 feedback-uri consecutive în aceeași direcție.
* Aplicația salvează aceste acțiuni într-o bază de date locală SQLite.

**Cerinte funcționale**:

* Interfață intuitivă (cu scale slider de la -3 la +3).
* Vizualizare istoric și comentarii.
* Salvarea înregistrărilor de feedback și login persistent.

**7.3 Proiectare detaliată de securitate**

Deoarece sistemul nu este conectat la internet și rulează local pe un calculator personal, riscurile externe sunt reduse. Totuși, sunt implementate măsuri de bază pentru protecția accesului la aplicație:

* Autentificare pe bază de username și parolă.
* Hasharea parolei (folosind SHA256).
* Verificarea parolei la login.
* Salvarea credentialelor în baza de date cu acces restricționat.

În viitor, se poate adăuga criptarea bazei de date și loguri pentru auditarea activității utilizatorului.

**7.4 Proiectare de detaliu pentru performanța sistemului**

Sistemul actual este optimizat pentru utilizare locală, cu un număr mic de utilizatori simultani. Pentru scenariul de testare, aplicația poate susține:

* 5–10 utilizatori locali (prin crearea de conturi).
* Răspuns rapid (<100ms) la interacțiuni (slider, trimitere feedback).
* Salvare instantanee în baza SQLite.

**Performanță estimată**:

* Feedback procesat în medie sub 1 secundă.
* Interfață responsive chiar pe sisteme modeste (CPU < 20% utilizare).

**7.5 Proiectare detaliată a comunicațiilor interne**

Toate componentele software comunică intern prin apeluri de funcții în Python. Feedbackul este transmis din GUI spre logică, apoi spre baza de date. Structura modulară permite înlocuirea componentelor sau extinderea cu module viitoare (actuatori, notificări, integrare online).

**Comunicări interne importante**:

* GUI → Controller: feedback și comentarii.
* Controller → Logica voturilor → Bază de date.
* Backend → GUI: actualizare valori simulate, afișare scoruri.

**8. Controale pentru verificarea integrității sistemului**

Sistemul inteligent de monitorizare a biroului implementează o serie de măsuri menite să asigure integritatea datelor, accesul controlat și auditabilitatea operațiunilor efectuate de utilizatori.

✔ **Acces controlat prin autentificare**  
Se implementează autentificarea pe bază de cont (utilizator și parolă hash-uită), pentru a restricționa accesul la funcționalitățile aplicației doar utilizatorilor înregistrați.

✔ **Limitarea accesului la datele critice**  
Doar utilizatorii autentificați pot vizualiza istoricul feedback-ului și pot trimite comentarii. Baza de date nu poate fi accesată direct din exterior, ci doar prin interfața aplicației.

✔ **Audit simplificat al feedback-ului**  
Fiecare vot trimis este asociat cu utilizatorul și timestamp-ul, permițând trasabilitatea modificărilor simulate asupra valorilor parametrilor (temperatură, lumină etc.).

✔ **Piste de audit pentru istoricul utilizatorului**  
Secțiunea „Istoric voturi” permite atât utilizatorului cât și administratorului să vizualizeze contribuțiile anterioare și să verifice dacă un anumit comportament a influențat parametrii din birou.

✔ **Validarea datelor introduse**  
Slider-ul permite doar valori între -3 și +3. Se previne introducerea unor valori eronate sau abuzive. Comentariile sunt verificate ca text (nu permit scripturi sau cod potențial periculos).

✔ **Protecție la ștergere sau modificări neautorizate**  
Datele sunt salvate automat în baza SQLite, fără posibilitate de ștergere din interfață. Accesul direct la baza de date este restricționat și este disponibil doar administratorului.

✔ **Identificare utilizator + dispozitiv** *(pentru versiuni viitoare)*  
Sistemul poate fi extins pentru a salva și adresa IP/localhost sau alte date despre terminalul de pe care s-a făcut logarea.