

**UNIVERSITATEA "ALEXANDRU-IOAN
CUZA" DIN IASI
FACULTATEA DE INFORMATICA**



LUCRARE DE LICENTA

**Aplicație pentru Optimizarea Proceselor de Producție,
Gestiune a Stocurilor și Management al Comenzilor în
Domeniul Prefabricatelor din Beton**

propusă de

ANDREI EDUARD

Sesiunea: FEBRUARIE, 2025

Coordonator științific

**IGNAT ANCA
UNIVERSITATEA "ALEXANDRU-IOAN
CUZA" DIN IAȘI
FACULTATEA DE INFORMATICA**

**Aplicație pentru Optimizarea Proceselor de Producție,
Gestiune a Stocurilor și Management al Comenzilor în
Domeniul Prefabricatelor din Beton**

ANDREI EDUARD

Sesiunea: FEBRUARIE, 2025

Coordonator științific

IGNAT ANCA

Avizat,

Indrumator lucrare de licenta,

IGNAT ANCA.

Data: 22.06.2023 Semnatura:

Declaratie privind originalitatea continutului lucrarii de licenta

Subsemnatul ANDREI EDUARD, domiciliat in Romania, jud. Neamt, sat Gheraesti, str. Aleea Teilor, nr. 14, nascut la data de 09.11.2000, identificat prin CNP 5001109270850, absolvent al Facultatii de Informatica, specializarea Informatica, promotia 2022, declar pe propria raspundere, cunoscand consecintele falsului in declaratii in sensul art. 326 din Noul Cod Penal si dispozitiile Legii Educatiei Nationale nr. 1/2011 art. 143 al. 4 si 5 referitoare la plagiat, ca lucrarea de licenta cu titlul "Aplicație pentru Optimizarea Proceselor de Producție, Gestiune a Stocurilor și Management al Comenzilor în Domeniul Prefabricatelor din Beton" , elaborata sub indrumarea doamnei IGNAT ANCA, pe care urmeaza sa o sustin in fata comisiei, este originala, imi apartine si imi asum continutul sau in intregime.

De asemenea, declar ca sunt de acord ca lucrarea mea de licenta sa fie verificata prin orice modalitate legala pentru confirmarea originalitatii, consimtind inclusiv la introducerea continutului ei intr-o baza de date in acest scop.

Am luat la cunostinta despre faptul ca este interzisa comercializarea de lucrari stiintifice in vederea facilitarii falsificarii de catre cumparator a calitatii de autor al unei lucrari de licenta, de diploma sau de disertatie si, in acest sens, declar pe proprie raspundere ca lucrarea de fata nu a fost copiata, ci reprezinta rodul cercetarii pe care am intreprins-o.

Data: 28.01.2025

Semnatura:

Anexa III

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul **„Aplicație pentru Optimizarea Proceselor de Producție, Gestiune a Stocurilor și Management al Comenzilor în Domeniul Prefabricatelor din Beton”**, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, 28.01.2025

Absolvent Andrei Eduard

(semnătura în original)

Anexa IV

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programelor-calculator, în format executabil și sursă, să aparțină autorului prezentei lucrări, **Andrei Eduard**.

Încheierea acestui acord este necesară din următoarele motive:

- Prezint intenția de a dezvolta în continuare lucrarea pentru a-mi întemeia, ulterior, o afacere care are ca obiect oferirea accesului la aplicația dezvoltată în cadrul lucrării

Iași,

Decan *Adrian Iftene* Absolvent *Andrei Eduard*

(semnătura în original)

CUPRINS

| | |
|---|-----------|
| Introducere | 7 |
| Motivatie | 8 |
| 1. Cercetare în vederea implementării..... | 10 |
| 1.1. SAGA (Soft de contabilitate românesc)..... | 10 |
| 1.2. Katana (Soluție modernă de gestiune a stocurilor și producției)..... | 10 |
| 1.3. Fishbowl (ERP și software de gestiune a inventarului)..... | 11 |
| 2. Concepte și tehnologii utilizate..... | 13 |
| 2.1. DESIGN UNIFORM..... | 13 |
| 2.2. REST..... | 13 |
| 2.3. JSON..... | 14 |
| 3. Descriere tehnica..... | 16 |
| 3.1 Alegerea tehnologiilor..... | 16 |
| 3.1.1. Python..... | 16 |
| 3.1.1.1. De ce Python?..... | 16 |
| 3.1.1.2. Pillow..... | 17 |
| 3.1.1.3. Django..... | 17 |
| 3.1.1.4. PyJWT..... | 18 |
| 3.1.2. Angular..... | 18 |
| 3.1.2.1. TypeScript..... | 19 |
| 3.1.2.2. Angular..... | 19 |
| 3.1.2.3. Ce este o aplicație single-page?..... | 19 |
| 3.1.3. PostgreSQL..... | 20 |
| 3.1.4. Electron..... | 21 |
| 4. Metodologia implementării..... | 23 |
| 4.1. Scheme de implementare..... | 23 |
| 4.1.1. Stack tehnologic..... | 23 |
| 4.1.2. Proiectarea bazei de date..... | 23 |
| 4.2. Backend..... | 24 |
| 4.2.1. Serverul Django..... | 24 |
| 4.3. Frontend..... | 26 |
| 4.3.1. Pagina de conectare..... | 27 |
| 4.3.2. Panel Admin (Prefabricate)..... | 28 |
| 4.3.3. Panel Admin (Clienți)..... | 30 |
| 4.3.4. Panel Admin (Utilizatori)..... | 31 |
| 4.3.5. Produse..... | 32 |
| 4.3.6. Pagină produs individual..... | 32 |
| 4.3.7. Pagina comenzi..... | 33 |
| 4.3.8. Pagina comandă activă..... | 35 |
| 4.3.9. Pagina & componentele calculator..... | 36 |
| 4.3.10. Comutator teme..... | 38 |
| 4.3.11. Interceptor HTTP..... | 38 |
| 4.3.12. Modulul de rutare..... | 40 |
| 4.3.13. Servicii..... | 40 |
| 4.3.14. Modele..... | 42 |
| 5. Concluzii..... | 44 |
| 6. Direcții de viitor..... | 45 |

INTRODUCERE

LUCRARE DE LICENTA, 2024, ANDREI EDUARD

Titlu: Aplicație pentru Optimizarea Proceselor de Producție, Gestione a Stocurilor și Management al Comenzilor în Domeniul Prefabricatelor din Beton

Descriere:

Această aplicație este proiectată pentru a simplifica procesul de stabilire a prețului de vânzare al produselor prefabricate din beton, integrând totodată funcționalități avansate pentru gestiunea stocurilor și evidența comenzilor. Utilizatorii vor putea introduce detalii tehnice despre diverse produse prefabricate, precum: tipul prefabricatului (bancă, coș de gunoi, bolard, cămin, spalier etc.), dimensiuni, tipul de armare și alte caracteristici relevante. Aplicația va calcula automat costurile de producție pe baza acestor informații.

În plus, aplicația va oferi:

1. **Gestiune de stocuri** – Aceasta va permite monitorizarea precisă a stocurilor și notificarea utilizatorilor în cazul în care acestea sunt insuficiente pentru a acoperi o comandă.
2. **Evidența comenzilor** – O comandă nouă poate fi introdusă în sistem, iar aplicația va verifica automat disponibilitatea produselor în stoc. După livrarea produselor, stocurile vor fi actualizate în mod automat, iar comanda va fi marcată ca finalizată.
3. **Autentificare și securitate** – Accesul în aplicație va fi gestionat printr-o pagină de login și creare cont, ce va include o parolă predefinită necesară pentru înregistrare și aprobarea ulterioară de către un administrator.

MOTIVAȚIE

Scopul aplicației:

Având o experiență de opt ani în domeniul prefabricatelor din beton, am identificat probleme recurente legate de gestiunea stocurilor și managementul comenzilor. Aplicația existentă în cadrul companiei întâmpină dificultăți în gestionarea corectă a stocurilor, ceea ce duce la discrepanțe și la întreruperi în activitate. În plus, există lipsuri în comunicarea dintre echipa de vânzări și cea responsabilă cu pregătirea comenzilor, ceea ce generează întârzieri și nemulțumiri din partea clienților.

Prin această aplicație, toate aceste probleme vor fi rezolvate, facilitând o gestiune eficientă a stocurilor, o comunicare mai bună între departamente și o organizare optimă a procesului de producție și livrare. Echipele responsabile de vânzări, pregătirea comenzilor și producție vor avea acces permanent la informații actualizate, ceea ce va contribui la creșterea productivității și satisfacției clienților.

1. Cercetare în vederea implementării

Problemele sistemelor de gestiune a stocurilor: **SAGA, Katana, Fishbowl**.

1.1. SAGA (Soft de contabilitate românesc)

Avantaje:

Este folosit pe scară largă în România, are suport pentru legislația locală și este accesibil ca preț.

Probleme:

Modulul de gestiune a stocurilor este destul de limitat comparativ cu soluțiile internaționale.

Nu este optimizat pentru e-commerce sau producție, fiind mai potrivit pentru firme mici.

Interfața este învechită și mai puțin intuitivă față de soluțiile moderne.

Sincronizare limitată cu alte softuri, ceea ce face dificilă integrarea cu platforme online sau ERP-uri avansate.

Cost:

Parțial gratuit – există o versiune gratuită pentru PFA-uri și ONG-uri, dar modulele mai avansate sunt contra cost.

1.2. Katana (Soluție modernă de gestiune a stocurilor și producției)

Avantaje:

Specializat pentru companii de producție și e-commerce.

Automatizare avansată a stocurilor și comenzilor.

Integrări cu Shopify, QuickBooks, Xero, WooCommerce etc.

Probleme:

Scump pentru firme mici (taxă lunară/subscripție).

Nu are suport direct pentru legislația românească.

Necesită o perioadă de învățare pentru utilizatori noi.

Cost:

Plătit – abonamente lunare începând de la aproximativ 100-200 €/lună, în funcție de funcționalități.

1.3. Fishbowl (ERP și software de gestiune a inventarului)

Avantaje:

Puternic pentru producție, distribuție și warehouse management.

Automatizări avansate, optimizat pentru firme mijlocii și mari.

Integrări cu QuickBooks, Amazon, Shopify etc.

Probleme:

Cost ridicat – mai potrivit pentru companii mari.

Complexitate mare, necesită training pentru utilizatori.

Greu de configurat pentru nevoile specifice ale firmelor mici.

Cost:

Plătit – licență inițială începând de la câteva mii de dolari + taxe anuale.

Concluzii cercetare

SAGA este singurul care are o variantă gratuită, dar este limitat.

Katana și Fishbowl sunt plătite, dar oferă mai multe funcționalități pentru producție și comerț online.

Dacă ai o firmă mică și operezi doar pe piața românească, SAGA poate fi suficient.

Pentru producție și vânzări online, Katana este o alegere mai modernă, iar pentru companii mari, Fishbowl este mai potrivit.

2. Concepte și tehnologii utilizate

2.1. DESIGN UNIFORM

În anul 2023, majoritatea aplicațiilor de succes care aveau o interfață web au respectat principiile Uniform Design. Aceste principii impun ca fiecare element al interfeței să îndeplinească aceleași reguli de funcționalitate și aspect.

Astfel, toate componentele unui sistem, precum texte, butoane, liste, câmpuri de input, ferestre de dialog și animații, trebuie să urmeze aceleași standarde de design. De exemplu:

- Toate butoanele de tip „Submit” trebuie să arate la fel pe toate paginile.
- Ferestrele de dialog cu funcționalități similare trebuie să aibă dimensiuni identice.
- Animațiile din interfață trebuie să aibă aceeași durată și același efect, folosind o curbă Bézier uniformă.
- Elementele de text cu același rol (titlu, descriere, text inactiv) trebuie să respecte aceleași reguli de culoare, dimensiune și grosime a fontului.

Uniform Design asigură consistență vizuală și funcțională, îmbunătățind experiența utilizatorului și ușurința de utilizare a aplicației.

2.2. REST

REST este un stil arhitectural utilizat pentru dezvoltarea de API-uri, construit peste protocolul HTTP (HyperText Transfer Protocol). Acesta permite accesul la metodele CRUD (Create, Read, Update, Delete) pentru gestionarea resurselor expuse de un API.

Comparativ cu SOAP (Simple Object Access Protocol), REST nu impune un standard strict de implementare, ceea ce îl face flexibil și compatibil cu orice limbaj de programare sau platformă. De asemenea,

REST simplifică accesul la resurse prin utilizarea unor URI-uri intuitive, evitând necesitatea definirii unui endpoint separat pentru fiecare acțiune.

Un API RESTful utilizează metodele HTTP standard pentru operarea asupra resurselor:

- **GET** `api/v1/users` – Returnează lista tuturor utilizatorilor.
- **POST** `api/v1/users` – Adaugă un utilizator nou.
- **PUT** `api/v1/users/{id}` – Actualizează un utilizator existent.
- **DELETE** `api/v1/users/{id}` – Șterge un utilizator.

În schimb, API-urile bazate pe SOAP folosesc endpointuri mai complexe, de exemplu:

- `api/v1/getUsers`
- `api/v1/addUser`

Din punct de vedere al securității, REST nu oferă un standard nativ de autentificare, fiind necesară implementarea unor soluții suplimentare, cum ar fi autentificarea prin sesiuni sau token-uri (ex: OAuth, JWT).

2.3. JSON

JSON este un format de date ușor de citit și utilizat, suportat de majoritatea limbajelor de programare. Principalul său scop este standardizarea reprezentării obiectelor pentru a facilita schimbul de date între sisteme diferite.

Fiind utilizat pe scară largă în comunicarea client-server, JSON permite ca un client web să primească și să proceseze date sub formă de obiecte, oferind astfel o integrare rapidă și eficientă între front-end și back-end.

Datorită simplității sale, JSON a devenit standardul principal pentru transferul de date în aplicațiile web moderne.

Un string JSON este delimitat fie de acolade {}, fie de paranteze pătrate [], în funcție de structura sa. Dacă JSON conține o listă de valori, atunci aceasta va fi reprezentată între paranteze pătrate.

JSON funcționează pe baza principiului cheie-valoare, permițând definirea atributelor pentru obiecte în limbajele orientate pe obiect sau utilizarea sa sub formă de dicționar (Hash Map) în limbajele care suportă astfel de structuri.

Un string JSON poate conține chei asociate cu diferite tipuri de valori, inclusiv liste, obiecte sau tipuri de date primitive, cum ar fi numere întregi, numere reale, valori booleene și șiruri de caractere. Datorită flexibilității sale, JSON este utilizat pe scară largă pentru schimbul de date între aplicații și sisteme diferite.

3. Descriere tehnica

Acest capitol are scopul de a descrie detaliile tehnice ale implementării platformei, motivele pentru care au fost alese anumite tehnologii, precum și modul de utilizare și integrare a conceptelor prezentate anterior.

3.1 Alegerea tehnologiilor

3.1.1. Python

Pentru implementarea conceptului de învățare automată și, implicit, a algoritmilor fundamentali folosiți în acest domeniu, am ales limbajul de programare Python.

Python este un limbaj de programare de nivel înalt, dinamic și interpretat, construit ca un wrapper peste limbajul C. Acesta oferă suport pentru mai multe paradigme de programare, inclusiv programarea orientată pe obiect (OOP).

3.1.1.1. De ce Python?

Printre principalele motive pentru care am ales Python în locul altor limbaje orientate pe obiect (Java, C++, C#) se numără:

- Simplitatea și ușurința cu care pot fi implementate diferite funcționalități.
- Documentația extinsă și bine structurată.
- Comunitatea vastă de dezvoltatori pasionați de statistică.
- Numărul mare de module open-source realizate de programatori, menite să facă dezvoltarea de aplicații mult mai facilă

De asemenea, Python permite implementarea facilă a unui API REST pentru expunerea funcționalităților algoritmilor, utilizând framework-uri precum **Django** sau **Flask**.

Pe lângă bibliotecile de bază incluse în Python, cele mai importante tehnologii utilizate în acest proiect sunt:

- **Pillow** – pentru procesarea și redimensionarea de imagini.
- **Django** – pentru dezvoltarea API-ului și gestionarea interacțiunii cu utilizatorii.

- **PyJWT** – pentru suportul oferit în vederea realizării modulului de autentificare de pe partea de back-end.

Alegerea Python și a ecosistemului său extins a permis dezvoltarea unei soluții flexibile și eficiente pentru implementarea platformei.

3.1.1.2. Pillow

Pillow (PIL - Python Imaging Library) este un modul Python specializat în prelucrarea de imagini. Oferă posibilitatea de a redimensiona, colora, edita, converti și edita datele imaginilor, suportând majoritatea formatelor utilizate în practică.

Am utilizat Pillow pentru a putea redimensiona cu ușurință imaginile produselor atât pentru a uniformiza procesul, dar mai ales pentru a salva spațiul de stocare din cloud, algoritmul de compresie utilizat de PIL putând să reducă dimensiunile imaginilor cu până la 80%.

3.1.1.3. Django

Django este un framework web de nivel înalt pentru Python, conceput pentru a permite dezvoltarea rapidă, sigură și eficientă a aplicațiilor. Acesta oferă suport pentru diverse implementări, valorificând cele mai recente tehnologii.

Fiind un framework axat pe securitate, Django oferă:

- Autentificare bazată pe sesiuni
- Criptarea parolelor utilizatorilor
- Sistem de permisiuni avansat, permițând alocarea de roluri și drepturi de acces pentru anumite endpoint-uri
- Crearea conturilor de superuser, cu acces complet la aplicație

Funcționalități cheie ale Django

Django oferă o serie de funcționalități care facilitează dezvoltarea și menținerea aplicațiilor:

- Migrarea automată a bazei de date
- Versionare automată a API-ului

- Crearea de microservicii (Django Applications) direct din linia de comandă
- Securizarea API-ului prin middlewares – utilizând JWT sau sesiuni
- Extensibilitate – module suplimentare precum drf-yasg pentru generarea automată a documentației API în Swagger sau JSON/YAML
- Scalare automată, adaptându-se la creșterea numărului de utilizatori

Am ales Django datorită avantajelor oferite, dar și experienței acumulate folosindu-l în proiecte universitare și personale.

3.1.1.4. PyJWT

PyJWT este un modul Python dedicat manipulării token-urilor JSON Web Tokens (JWT), esențial în implementarea sistemelor moderne de autentificare și autorizare. Acest modul permite generarea, semnarea, decodarea și validarea token-urilor, facilitând o metodă sigură și eficientă de a gestiona accesul utilizatorilor la resursele protejate ale aplicațiilor. PyJWT suportă o gamă variată de algoritmi criptografici, precum HS256, RS256 și ES256, adaptându-se astfel la cerințele specifice de securitate ale fiecărui proiect. Prin integrarea PyJWT, se reduce semnificativ complexitatea codului de autentificare, contribuind la crearea unui sistem robust de gestionare a sesiunilor și la protejarea datelor sensibile împotriva accesului neautorizat.

3.1.2. Angular

Pentru a înțelege mai bine Angular, trebuie să definim câțiva termeni esențiali:

- **HTML (HyperText Markup Language)** – limbaj de marcare utilizat pentru definirea structurii paginilor web.
- **SCSS (Sassy Cascading Style Sheets)** – o versiune avansată de CSS, care oferă:
 - Variabile pentru reutilizarea stilurilor
 - Mixins pentru definirea proprietăților reutilizabile
 - Nesting, care permite organizarea mai eficientă a stilurilor prin moștenire

În continuare, vom detalia funcționalitățile Angular și modul în care acesta facilitează dezvoltarea interfețelor web moderne.

3.1.2.1. TypeScript

TypeScript este un superset al limbajului JavaScript, creat pentru a adăuga tipuri de date pentru variabile, astfel încât să evite erorile și să ofere dezvoltatorilor un control mai precis asupra fluxului codului. Pe lângă aceasta, TypeScript adaugă și alte funcționalități care nu sunt prezente în JavaScript, cum ar fi:

- Interfețe pentru abstractizare
- Suport pentru decoratori
- Tipuri generice
- Inferența tipurilor

Codul scris în TypeScript este mai ușor de întreținut și dezvoltat datorită capacității IDE-urilor de a detecta erorile de tip la compilare, spre deosebire de JavaScript, unde erorile de tip sunt descoperite doar la runtime. De asemenea, debugging-ul în TypeScript este mai ușor, deoarece un debugger poate oferi sugestii și moduri diferite de vizualizare a variabilelor pe baza tipurilor acestora.

3.1.2.2. Angular

Angular este un framework de dezvoltare și design pentru aplicații, creat de Google începând cu 2010. Scopul său principal este să permită dezvoltarea de aplicații single-page eficiente și sofisticate, într-un mediu ușor de înțeles, cu o curbă de învățare mică. Angular folosește o arhitectură orientată pe obiecte, similară cu cele din limbaje precum C# sau Java.

3.1.2.3. Ce este o aplicație single-page?

O aplicație single-page este o aplicație web care încarcă o singură pagină HTML și actualizează doar conținutul acesteia, fără a reîncărca întreaga pagină atunci când utilizatorul navighează între secțiuni.

În Angular, fiecare parte a aplicației reprezintă o componentă, care este alcătuită dintr-un fișier TypeScript, un model HTML și un fișier

CSS/SCSS asociat. Componentele sunt integrate într-un fișier HTML principal care reprezintă baza aplicației.

Comutarea între componente se face printr-un sistem de rutare oferit de Angular, care asociază componentele cu scheme URL, iar utilizatorul rămâne pe aceeași pagină.

Angular se bazează pe TypeScript și profită de capacitățile acestui limbaj pentru a oferi un suport avansat pentru dezvoltatori și a face implementările complexe mult mai ușor de scris.

3.1.3. PostgreSQL

PostgreSQL este un sistem de baze de date relațional open-source, cu peste 25 de ani de dezvoltare continuă. Este una dintre cele mai populare implementări, având suport nativ în majoritatea limbajelor de programare. PostgreSQL se axează pe scalabilitate, interoperabilitate și posibilitatea de a utiliza tipuri de date comune cu limbajele care o folosesc, oferind, de asemenea, numeroase îmbunătățiri de tip "Quality of Life" (QoL), printre care:

- Posibilitatea de a adăuga sau elimina coloane dintr-un tabel fără a fi necesară copierea acestuia în întregime
- Utilizarea SQL standard, ceea ce asigură interoperabilitatea între majoritatea sistemelor
- Implementări ORM (Object Relational Mapper) în limbajele populare (Python, Java, C++, Rust, etc.)
- Costuri reduse de întreținere pentru baze de date mari
- Securitate robustă, fiind un proiect open-source
- Suport pentru tipuri de date nestructurate (imagini, audio, video)
- Procesare concurentă a query-urilor fără deadlock în 99.9999% din cazuri
- Suport nativ în IDE-uri

Am ales PostgreSQL în detrimentul altor sisteme de gestionare a bazelor de date datorită ușurinței în editarea tabelelor, suportului din IDE-uri și posibilității de a lucra cu tipuri de date nestructurate. De asemenea, recomandarea Django de a utiliza PostgreSQL este un alt

factor important, având în vedere că ORM-ul Django este optimizat pentru acest sistem.

3.1.4. Electron

Electron este un framework open-source care permite dezvoltarea de aplicații desktop utilizând tehnologii web, cum ar fi HTML, CSS și JavaScript. Acesta combină puterea Chromium pentru renderizarea interfețelor grafice și Node.js pentru a permite aplicațiilor să aibă acces la funcționalitățile native ale sistemului de operare. Electron oferă o platformă robustă și versatilă pentru crearea de aplicații multi-platformă (Windows, macOS, Linux) cu un singur set de cod.

Printre avantajele oferite de Electron se numără:

- Dezvoltare cross-platform: Permite crearea de aplicații care rulează pe toate cele trei platforme majore (Windows, macOS, Linux) folosind aceleași tehnologii web, fără a fi nevoie de modificări semnificative ale codului.
- Acces la API-uri native: Electron combină puterea Node.js cu interfața grafică oferită de Chromium, oferind acces direct la funcționalitățile native ale sistemului de operare, precum fișierele locale, clipboard, notificări și multe altele.
- Suport pentru interfețe grafice complexe: Deși Electron folosește HTML și CSS pentru interfețele grafice, poate implementa interfețe sofisticate, de la cele mai simple la cele mai avansate, cu un control detaliat asupra stilizării și funcționalității.
- Actualizări automate: Electron permite actualizarea aplicațiilor desktop în mod simplu, cu opțiuni integrate pentru actualizări automate, reducând efortul de întreținere.
- Simplitate în implementare: Folosind JavaScript și librării web existente, dezvoltatorii nu trebuie să învețe un limbaj de programare complet nou pentru a construi aplicații desktop, iar ecosistemul vast de librării JavaScript și npm permite extinderea rapidă a funcționalității aplicației.

Am ales Electron pentru dezvoltarea aplicațiilor desktop datorită flexibilității sale de a crea aplicații compatibile cu multiple platforme, ușurinței de integrare cu tehnologii web existente și capacității de a

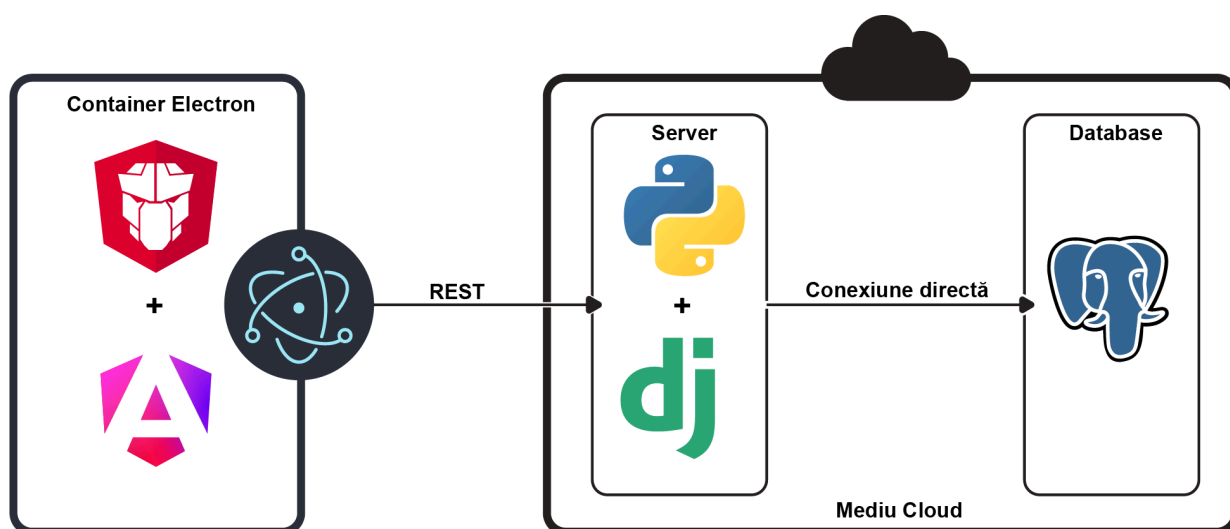
gestiona atât interfețele grafice cât și accesul la funcționalitățile native ale sistemului. Acest framework este deosebit de util atunci când se dorește crearea rapidă de aplicații desktop puternice, cu un control complet asupra experienței utilizatorului și performanței aplicației.

4. Metodologia implementării

Pentru realizarea aplicației, am utilizat o metodă apropiată de metodologia Agile, dar fără a include board-uri sau sprint-uri, dezvoltând aplicația fără a lucra într-o echipă. Totuși, m-am folosit de o listă de To-Dos pentru a-mi organiza task-urile, eliminând elementele din listă abia după ce eram sigur că funcționalitățile descrise au fost verificate și funcționează corespunzător.

4.1. Scheme de implementare

4.1.1. Stack tehnologic



După cum se observă din figura de mai sus, am utilizat Angular și PrimeNG pentru realizarea aplicației single page propriu-zisă, fiind inclusă apoi într-un container Electron. Aceasta comunică prin REST cu server-ul, realizat cu Django și Python, care la rândul său comunică printr-o conexiune directă cu baza de date PostgreSQL. Atât serverul cât și baza de date sunt ținute într-un mediu cloud, pentru a putea face aplicația portabilă fără nicio dependență și pentru a putea realiza sincronizarea datelor pentru toți utilizatorii.

4.1.2. Proiectarea bazei de date

Pentru a putea coordona toate tipurile de date utilizate în aplicație, a fost necesară proiectarea bazei de date prin crearea de modele Django pentru fiecare obiect ce urma a fi utilizat. Fiecare model a fost creat cu

Având atât de multe atribuții, arhitectura acestuia nu putea fi monolitică, dat fiind faptul că de la începutul dezvoltării s-a dorit un server modular, capabil de a fi scalat; am ales astfel o arhitectură bazată pe servicii.

View-urile sunt realizate în fișierele ce conțin *views* în nume, utilizând clase ce moștenesc `GenericAPIView` pentru a fi implementate, respectând astfel best practice-ul Django.

Serviciile oferite de server sunt:

- *Authentication*

Acest serviciu se ocupă de creare de conturi, validare de JWT, logare și delogare utilizatori și adăugare de blacklist a token-urilor deja utilizate pentru conectare pentru a spori securitatea.

Implementarea utilizează `rest_framework_simplejwt` pentru a asigura căre fiecare utilizator o pereche de token-uri care sunt trimise înapoi la fiecare request făcut de utilizator către server.

Aici sunt oferite următoarele endpoint-uri:

- `auth/register/` - înregistrare utilizator
- `auth/token/` - logare
- `auth/token/verify/` - verificare utilizator conectat
- `auth/token/refresh/` - actualizare token expirat
- `auth/master/change/` - actualizare parolă pentru creare cont
- `auth/admin/` - pentru a adăuga/șterge un utilizator ca admin

- *Main*

Acest serviciu conține view-uri pentru toate funcționalitățile principale ale aplicației, nefiind posibilă separarea acestora pentru modularitate (ar elimina funcționalitatea dorită). Există view-uri pentru categorii de produse, clienți, comenzi, produse, fiecare permițând adăugare, modificare, ștergere, preluare și alte funcționalități conexe pentru fiecare categorie.

- *Stats*

Acest serviciu conține view-uri pentru statisticile aplicației, permițând utilizatorilor să apeleze endpoint-uri pentru a genera statistici generale (cel mai vândut produs, număr de comenzi și comenzi finalizate, cea mai mare comandă, top 10 clienți după numărul de comenzi și suma cheltuită) sau pentru statistici specifice

- produse
 - cantitate vândută
 - sumă vândută
 - cele mai populare produse
- comenzi
 - număr comenzi/perioadă de timp
 - sume cheltuite în comenzi și venituri/perioadă de timp

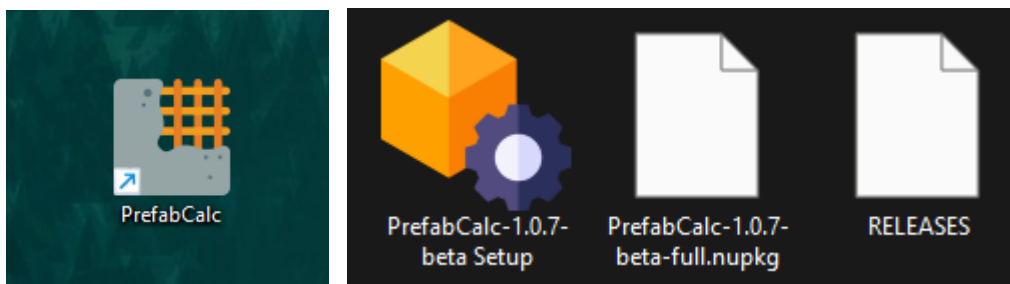
4.3. Frontend

Partea de frontend este realizată utilizând framework-ul de frontend Angular, alături de biblioteca de componente PrimeNG pentru a realiza interfața și a oferi componente care respectă standardele de accesibilitate.

Arhitectura este modulară iar fiecare componentă reprezintă o parte minimă care poate funcționa independent față de restul; acest fapt face atât integrarea de noi funcționalități, cât și eliminarea lor, foarte simplă.

Pentru comunicare cu backend-ul s-au utilizat RxJS și modulul integrat HTTP din Angular, HttpClient. Comunicarea cu partea de backend este realizată în servicii separate de componente, astfel atingând principiul de Single Responsibility.

Pentru a face aplicația portabilă am încapsulat toată funcționalitatea într-o aplicație desktop, utilizând Electron și Electron Forge (squirrel packager). Astfel, pentru a distribui aplicația, utilizatorul trebuie doar să ruleze installer-ul, iar aplicația va fi adăugată pe desktop-ul său și va fi gata de utilizare, fără a fi nevoie de alte dependențe. De asemenea, pentru procesul de update, utilizatorul trebuie doar să instaleze un nou setup, iar aplicația va fi înlocuită de versiunea nouă, păstrând în același timp datele utilizatorului.



În imaginile de mai sus sunt prezentate shortcut-ul realizat pe desktop după instalare, cât și fișierele generate de Electron Forge. (pentru instalare este suficientă existența fișierului de Setup, fără a fi necesare fișierele .nupkg sau RELEASES)

4.3.1. Pagina de conectare

The screenshot shows the PrefabCalc application window with two main sections: 'Înregistrare' (Registration) and 'Conectare' (Login).

Înregistrare (Registration):

- Text: 'Introdu informațiile mai jos' (Enter the information below)
- Fields: Nume utilizator (Username), E-mail, Parolă (Password), Confirmare parolă (Confirm password), Parolă Master (Master password).
- Button: Înregistrare (Register)

Conectare (Login):

- Text: 'Completează câmpurile de mai jos' (Fill in the fields below)
- Fields: Nume utilizator (Username), Parolă (Password).
- Button: Conectare (Login)

Pagina de conectare oferă utilizatorului posibilitatea de a se conecta sau de a își crea un cont nou. Pentru securitate, pentru crearea unui cont nou este necesară și introducerea unei parole master, setată de administratorul aplicației.

4.3.2. Panel Admin (Prefabricate)

The screenshot displays the 'Prefabricate' admin panel. At the top, there are navigation links for 'Calculator', 'Produse', and 'Comenzi'. A user profile 'Hei, admin' is visible in the top right. On the left, a sidebar menu lists 'Aplicație', 'Prefabricate', 'Clienți', and 'Utilizatori'. The main area is titled 'Prefabricate' and includes a '+ Aduagă' button. Below this is a search bar 'Caută...' and a 'Mod vizionare' button. A table lists products with columns for 'Cod Produs', 'Nume', 'Descriere', 'Status', 'Stoc', 'Preț', and 'Categorie'. The table contains three rows of product data.

| Cod Produs | Nume | Descriere | Status | Stoc | Preț | Categorie |
|------------|----------------------------|----------------|-------------|------|--------|----------------|
| 15 | camin 140x140x15 | test | STOC SCĂZUT | 500 | 500.00 | Camine rectang |
| 9 | CAMIN RECTAN 60x80x 100x10 | CAMIN | ÎN STOC | 9900 | 450.00 | Camine rectang |
| 13 | SPALIE R 210X 7X8 | SPALIER 210X7X | ÎN STOC | 500 | 29.00 | Spalieri |

După conectarea cu un cont de admin, utilizatorul este trimis către panel-ul de administrare. Acesta conține un meniu pentru management de prefabricate, clienți și utilizatori.

Partea de management prefabricate conține un tabel ce are funcționalități de căutare, sortare, ascundere și redimensionare coloane, vizualizare imagini în format full-screen și moduri de vizionare și editare.

Modul de vizionare permite vizualizarea tuturor produselor și a imaginilor acestora (în cazul în care un produs nu are imagine, aceasta este înlocuită dinamic cu o imagine cu numele produsului prin crearea unui URL personalizat); modul de editare permite editarea tuturor câmpurilor produselor.

Tot această pagină permite utilizatorilor să creeze noi produse în 3 pași, specificând categoria, numele, descrierea, stocul, codul (opțional), imaginea (opțional) și calculând dinamic prețul (componentele funcționale calculator).

Adaugă

1

Categorie

Alege o categorie

Camine rotunde X v

Mai departe

2

Detalii

3

Cost & preț

X Anulează

Salvează

PrefabCalc

Adaugă

2

Detalii

Introdu detaliile produsului

Nume

Descriere

0 buc


Cod produs (va fi setat automat dacă nu se completează)

Încarcă o imagine pentru produs (opțional)

+ Alege

Încarcă

X Anulează



X Anulează

Salvează

Mai sus sunt prezentate imagini cu procesul de adăugare a produselor noi. Se poate observa că dialogul pentru adăugarea produsului poate fi dimensionat astfel încât să ocupe întreg ecranul (în special util pentru ecrane mici).

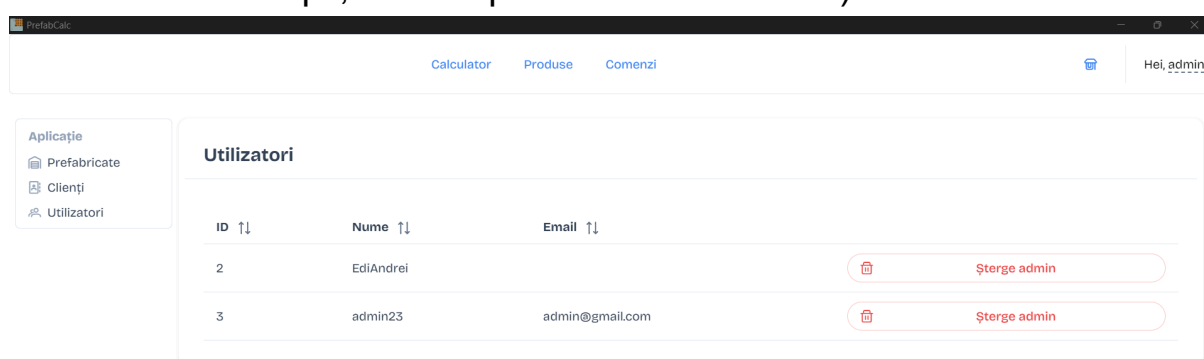
4.3.3. Panel Admin (Clienți)

Componenta de management clienți conține, precum cea pentru produse, un tabel în care se poate căuta, sorta, redimensiona, alături de un buton de "Adaugă".

Dialogul pentru adăugarea clienților conține toate datele clientului: denumire, adresă, telefon, e-mail, bancă, IBAN, CUI și număr înregistrare. Toate datele pot fi editate, iar clienții pot fi șterși din baza de date în modul editare.

4.3.4. Panel Admin (Utilizatori)

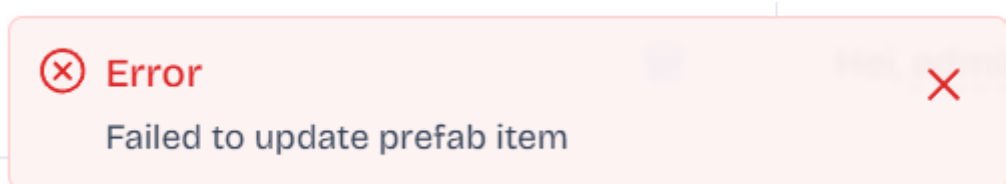
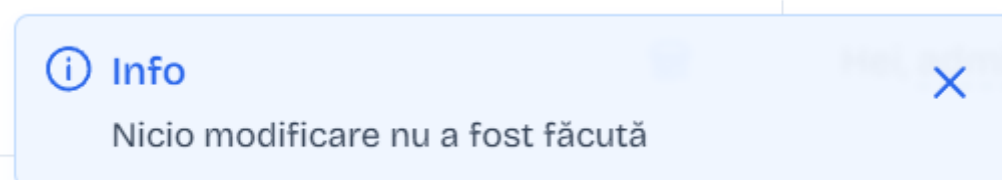
Această componentă conține un tabel cu utilizatori (ID, Nume și Email), alături de un buton de adaugă/șterge admin, care oferă sau șterge drepturile de administrator pentru un utilizator (la înregistrare, utilizatorii sunt utilizatori simpli, fără drepturi de administrator).



The screenshot shows the 'PrefabCalc' application window. The top navigation bar includes 'Calculator', 'Produse', and 'Comenzi'. A sidebar on the left lists 'Aplicație', 'Prefabricate', 'Clienți', and 'Utilizatori'. The main content area is titled 'Utilizatori' and contains a table with columns 'ID', 'Nume', and 'Email'. There are two rows of users. Each row has a 'Șterge admin' button with a trash icon.

| ID | Nume | Email | |
|----|-----------|-----------------|--------------|
| 2 | EdiAndrei | | Șterge admin |
| 3 | admin23 | admin@gmail.com | Șterge admin |

Toate acțiunile din aplicație sunt însoțite de mesaje de tip Toast (care dispar după câteva secunde) pentru a oferi un feedback utilizatorului pentru acțiunea realizată. Acestea sunt de tip Succes, Info sau Eroare, având un titlu scurt și un mesaj.

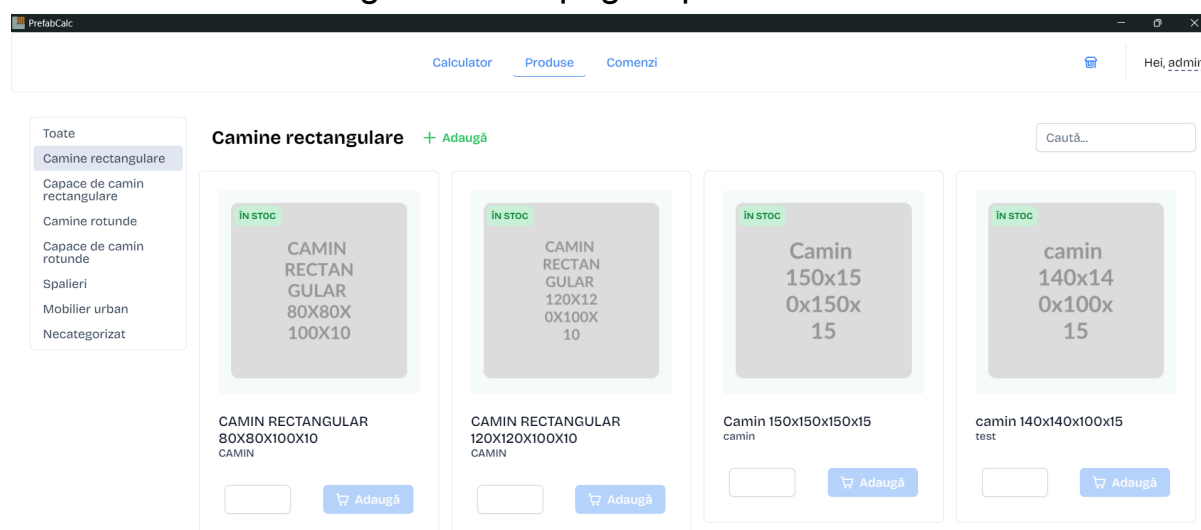


4.3.5. Produse

Pagina produse conține un meniu ce permite utilizatorilor să aleagă categoria pe care să o vizualizeze (sau să vizualizeze toate produsele). De asemenea, conține și un câmp pentru a căuta un produs după nume sau ID (filtrarea este realizată pe partea de frontend, lista produselor fiind încărcată în întregime la deschiderea paginii). Mai mult, pagina fiecărei categorii permite adăugarea unui nou produs în categoria respectivă, nefiind necesară navigarea către Panel Admin de fiecare dată.

Fiecare produs poate fi adăugat în comanda curentă prin completarea câmpului de sub imaginea sa și apăsarea pe butonul de "Adaugă".

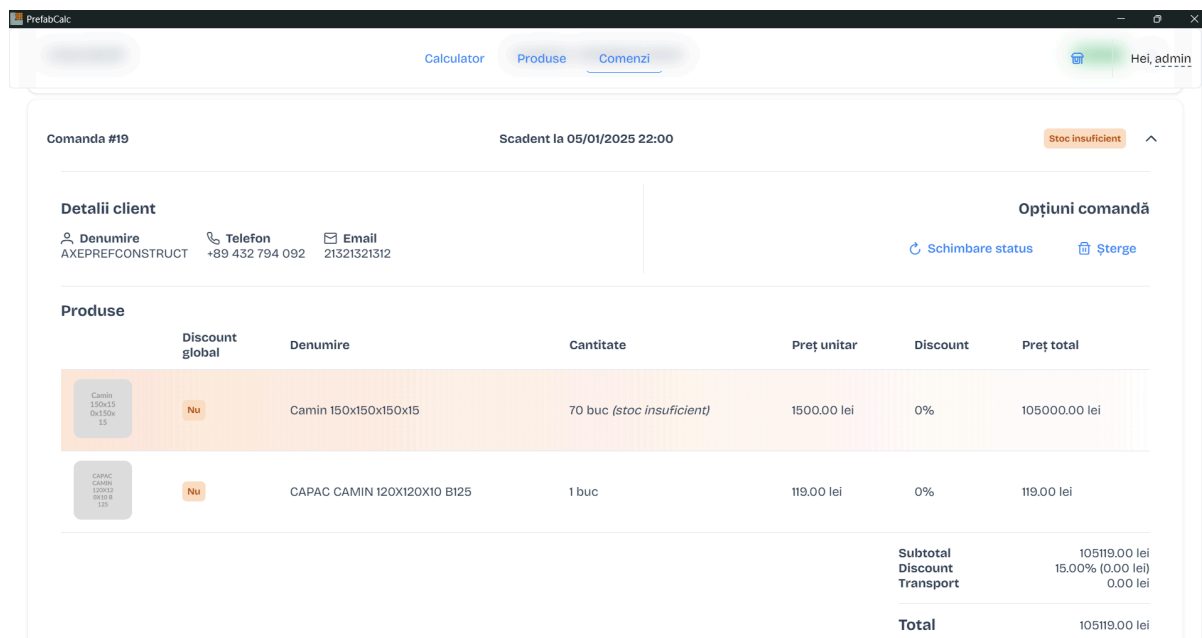
Fiecare produs conține și un tag care indică statusul pentru stocul său, nefiind necesară navigarea către pagina produsului.



4.3.6. Pagină produs individual

La apăsarea pe numele unui produs utilizatorul este redirecționat către pagina individuală a produsului, unde sunt prezente mai multe informații, precum cod produs, descriere, categorie, dimensiuni, cost, preț și istoric stoc.

Fiecare comandă are afișat numărul, data scadenței și statusul, iar la apăsarea pe o comandă utilizatorului îi sunt afișate detalii despre aceasta.



Comanda #19 Scadent la 05/01/2025 22:00 Stoc insuficient



Detalii client

Denumire: AXEPREFCONSTRUCT Telefon: +89 432 794 092 Email: 21321321312

Opțiuni comandă

Schimbare status Șterge

Produse

| | Discount global | Denumire | Cantitate | Preț unitar | Discount | Preț total |
|---|-----------------|-----------------------------|---------------------------|-------------|----------|-------------------|
|  | Nu | Camin 150x150x150x15 | 70 buc (stoc insuficient) | 1500.00 lei | 0% | 105000.00 lei |
|  | Nu | CAPAC CAMIN 120X120X10 B125 | 1 buc | 119.00 lei | 0% | 119.00 lei |
| Subtotal | | | | | | 105119.00 lei |
| Discount | | | | | | 15.00% (0.00 lei) |
| Transport | | | | | | 0.00 lei |
| Total | | | | | | 105119.00 lei |

Detaliile comenzii includ detalii despre client, opțiuni comandă (schimbare status pentru comenzile care nu sunt finalizate și ștergere pentru orice comandă), alături de o listă de produse și un breakdown pentru cost (subtotal, discount, transport).

Lista de produse conține imaginea, discount-ul global, denumirea, cantitatea, prețul unitar, discount-ul și prețul final; fiecare produs care nu are stoc suficient pentru finalizarea comenzii este afișat cu un fundal specific pentru a fi mai ușor de observat.

O comandă ce are statusul "Stoc insuficient" nu poate fi schimbată într-o comandă cu status "Finalizată", server-ul returnând o eroare care conține lista de produse cu stoc insuficient din cauza cărora nu poate fi finalizată comanda.



Info

Nu se poate finaliza comanda. Stoc insuficient pentru Camin 150x150x150x15.

4.3.8. Pagina comandă activă

Pagina comandă activă conține lista de produse din comanda curentă, opțiuni pentru alegerea clientului, a datei de scadență, discount-ului și costului de transport.

Fiecare produs din listă poate fi editat din punct de vedere al cantității, discount-ului, setării discount-ului global sau eliminarea produsului.

Această pagină include și dialoguri pentru adăugarea produselor la comandă fără a fi necesară navigarea către pagina cu toate produsele sau către pagina de produs individual, permițând de asemenea și adăugarea de noi produse în baza de date care pot fi apoi adăugate la comandă, nefiind necesară navigarea către Panel Admin → Management Prefabricate pentru a adăuga noi produse.

The screenshot displays the 'Comandă activă' (Active Order) page. At the top, there are navigation links: 'Calculator', 'Produse', and 'Comenzi'. The main section is titled 'Comandă activă' and includes a '+ Adaugă produs' button and a 'Renunță' button. Below this is a table of products in the order:

| Discount global | Denumire | Cantitate | Preț unitar | Discount | Preț total |
|-------------------------------------|---------------------------------------|-----------|-------------|----------|-------------|
| <input checked="" type="checkbox"/> | SPALIER 210X7X8 | 10 buc | 29.00 lei | 10 % | 261.00 lei |
| <input type="checkbox"/> | banca din beton 120x40x40, tip Marcos | 5 buc | 650.00 lei | 0 % | 3250.00 lei |

On the right, the 'Detalii' (Details) section shows:

- Date client: AXEPREFCONSTRUCT
- AXEPREFCONSTRUCT | FSDPG434SFSDP
- 2132132132
- +89 432 794 092

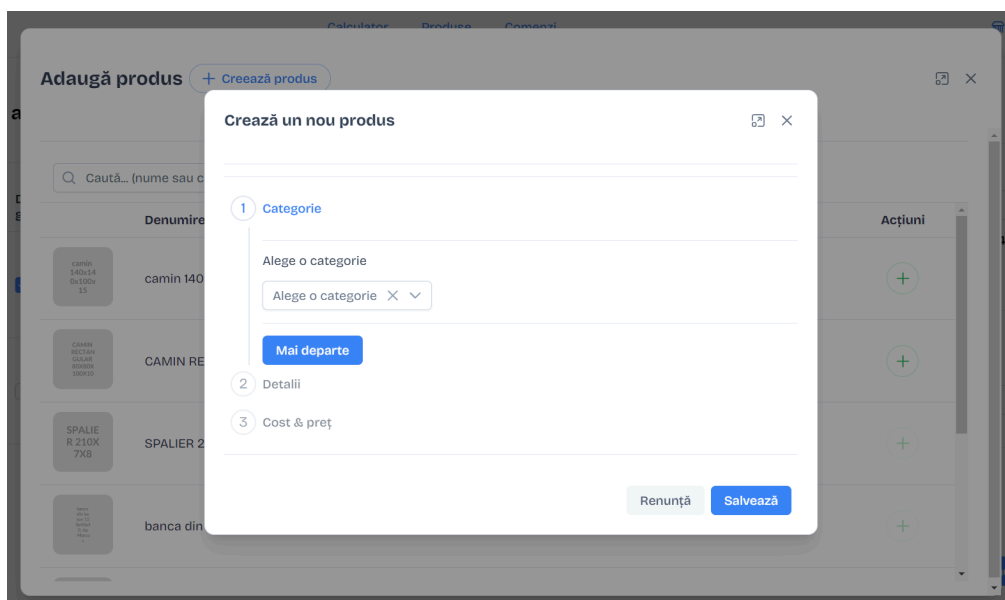
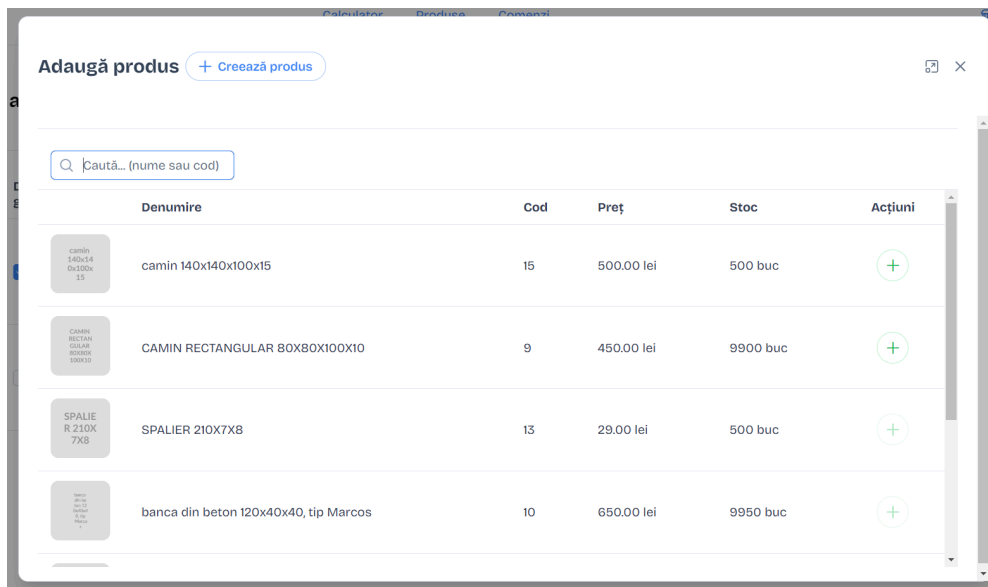
Below this, the 'Detalii comandă' (Order details) section shows:

- Data scadență: 05.12.2024
- Discount: 7 %
- Transport: 650 lei

At the bottom, the summary section shows:

- Subtotal: 3511.00 lei
- Discount: -18.27 lei
- Transport: +650 lei
- Total: 4142.73 lei

A 'Finalizează comanda' button is at the bottom right.



Pagina de comandă activă oferă și opțiunea de a renunța la întreaga comandă, eliminând toate produsele existente în comandă. Comanda curentă este salvată între sesiuni cât timp utilizatorul este conectat folosind `localStorage`.

4.3.9. Pagina & componentele calculator

Pagina calculator oferă o listă (sau un grid) ce conține toate tipurile principale de produse (produse prestabilite): Cămin rectangular, capac cămin rectangular, cămin rotund, capac cămin rotund și spalier.

La apăsarea butonului "Calculează" se deschide un panel care conține componentele funcționale calculator ce realizează automat costul produsului ales, utilizatorul specificând doar dimensiunile și eventuale opțiuni, alături de manoperă.

Pagina calculator oferă utilizatorilor un mod de calcul în scop informativ pentru costurile de realizare a diferitelor produse, putând fi utilizată în timpul convorbirilor cu eventuali clienți pentru a putea realiza pe loc prețuri pentru aceștia.

The screenshot displays the 'PrefabCalc' application interface. On the left, a sidebar lists four products: 'Cămin Rectangular', 'Capac cămin rectangular', 'Cămin Rotund', and 'Capac cămin rotund', each with a small image and 'IN STOC' status. The main area on the right is titled 'Cămin Rectangular' and contains the following fields:

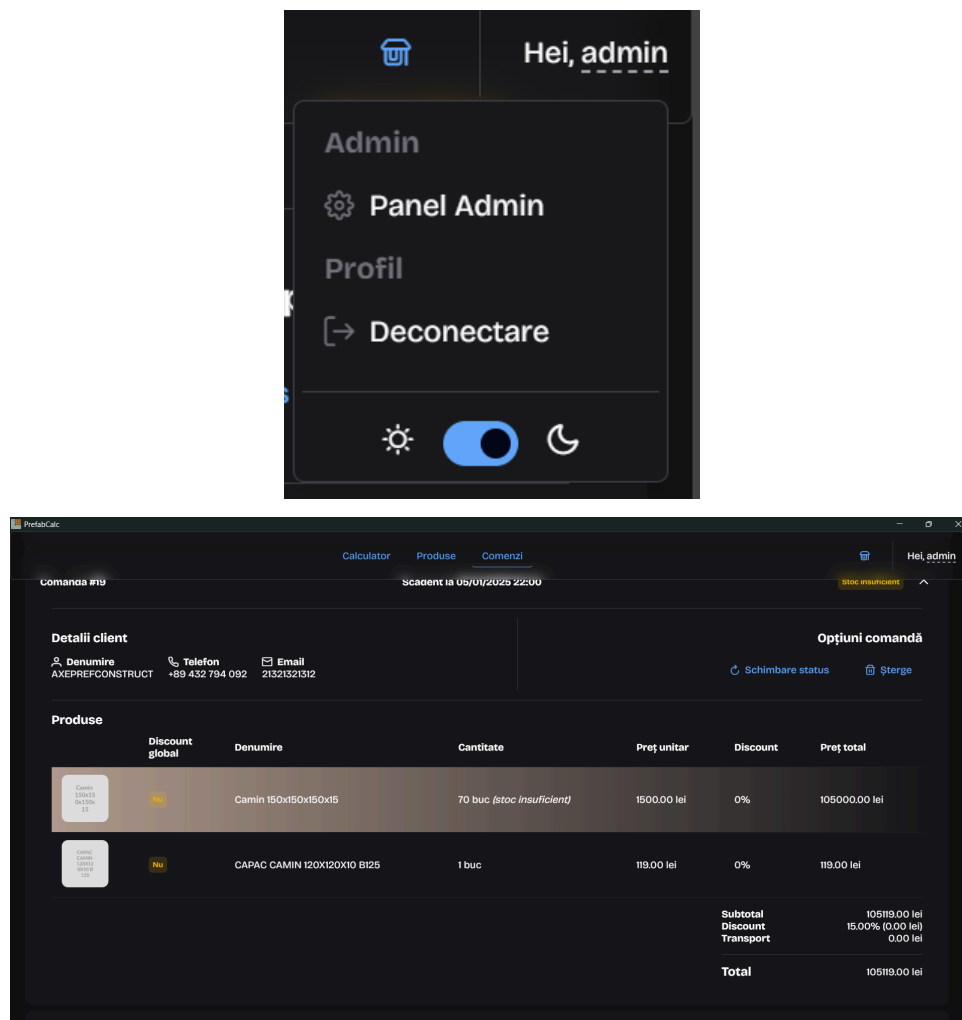
- Dimensiuni:**
 - Lungime: 0.10 m
 - Lățime: 0.10 m
 - Înălțime: 0.10 m
 - Grosime: 0.10 m
- Extra:**
 - Armare: Alege un tip de armare (dropdown)
 - Necesar fier: 0.00
 - Opțiuni cămin vane: ☐ Cămin de vane
 - Număr Tevi PVC: 0
 - Manoperă: 5 ore
- Total:** 139.15 RON
- Buttons:** 'Resetează' (red)

Componentele calculator sunt utilizate atât în scop informativ, cât și pentru ca utilizatorii să adauge produse noi cu dimensiuni specifice, nemaifiind necesară calcularea în prealabil a costurilor de realizare pentru produse, acestea fiind adăugate automat direct din aplicație. Se elimină astfel nevoia de a avea încă un program pentru acest calcul sau utilizarea unui pix și a hârtiei pentru realizarea calculelor.

Se elimină astfel și eroarea umană de a calcula costuri pentru un număr mare de produse, cât și efortul de a face aceste calcule pentru a realiza o listă sau o ofertă de produse.

4.3.10. Comutator teme

Pentru a satisface nevoile oricărui utilizator, aplicația oferă și posibilitatea unei teme închise, oferind un comutator în meniul utilizatorului



4.3.11. Interceptor HTTP

Deoarece autentificarea și autorizarea sunt realizate folosind JWT a fost necesară adăugarea unui interceptor HTTP pentru partea de frontend care să realizeze, înainte de trimiterea de request-uri către backend, adăugarea token-ului de acces în header-ul request-ului.

```

intercept(request: HttpRequest<any>, next: HttpHandler) {
  if (this.getAccessToken() !== '') {
    request = this.addToken(request, this.getAccessToken());
  }

  return next.handle(request).pipe(
    catchError((error) => {
      if (error instanceof HttpResponse && error.status === 401) {
        return this.handle401Error(request, next);
      } else {
        throw error;
      }
    })
  );
}

```

```

private handle401Error(request: HttpRequest<any>, next: HttpHandler) {
  if (!this.isRefreshing) {
    this.isRefreshing = true;
    this.refreshTokenSubject.next(null);

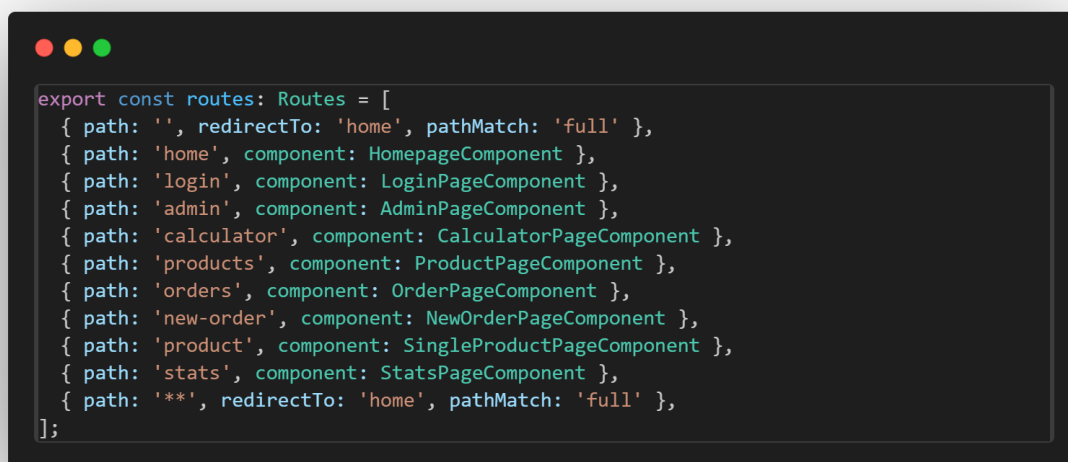
    return this.authService.refreshToken().pipe(
      switchMap((res: any) => {
        const token = res.access;
        this.isRefreshing = false;
        this.refreshTokenSubject.next(token);
        localStorage.setItem('access_token', token);
        return next.handle(this.addToken(request, token));
      })
    );
  } else {
    return this.refreshTokenSubject.pipe(
      filter((token) => token != null),
      take(1),
      switchMap((jwt) => {
        return next.handle(this.addToken(request, jwt));
      })
    );
  }
}

```

De asemenea, în cazul în care token-ul de acces este expirat, interceptorul se ocupă și de trimiterea unui token de refresh către server și înlocuirea token-ului vechi de acces cu unul nou, trimițând apoi request-ul inițial din nou către server. Astfel, utilizatorul nu primește nicio eroare iar cererea sa este realizată fără să fie nevoie să acționeze din nou în acest scop, după cum este prezentat mai sus.

4.3.12. Modulul de rutare

Modului de rutare este oferit de framework și face legătura dintre un URL și afișarea unei componente specifice pentru a simula astfel activitatea unui site cu mai multe pagini, păstrând totuși funcționalitatea într-un Single Page Application.



```
export const routes: Routes = [
  { path: '', redirectTo: 'home', pathMatch: 'full' },
  { path: 'home', component: HomepageComponent },
  { path: 'login', component: LoginPageComponent },
  { path: 'admin', component: AdminPageComponent },
  { path: 'calculator', component: CalculatorPageComponent },
  { path: 'products', component: ProductPageComponent },
  { path: 'orders', component: OrderPageComponent },
  { path: 'new-order', component: NewOrderPageComponent },
  { path: 'product', component: SingleProductPageComponent },
  { path: 'stats', component: StatsPageComponent },
  { path: '**', redirectTo: 'home', pathMatch: 'full' },
];
```

Ultimul path din imaginea de mai sus reprezintă orice URL nu este prezent în listă, redirecționând utilizatorul către pagina de acasă oricând acesta ajunge pe o pagină inexistentă.

4.3.13. Servicii

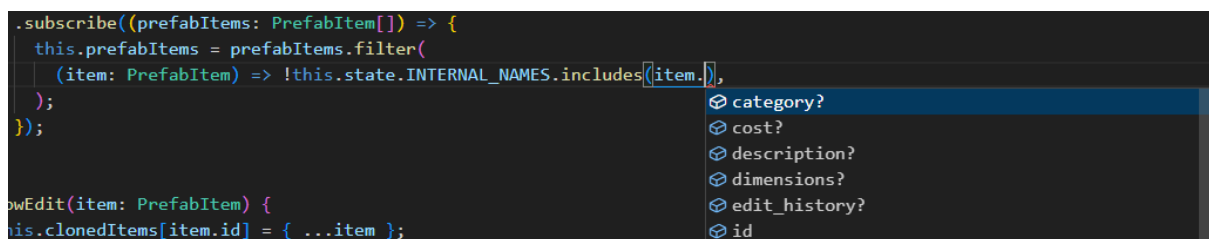
Comunicarea cu partea de backend a aplicației este realizată prin servicii Angular. Fiecare serviciu are un singur rol și îndeplinește o singură funcționalitate, fiind posibilă astfel mutarea unui serviciu sau rescrierea sa fără ca aceasta să afecteze celelalte servicii sau componente conexe. Toate serviciile sunt instanțiate o singură dată la deschiderea aplicației și se utilizează această unică instanță a acestora, indiferent de numărul de componente care le utilizează.

- *Auth Service*
 - autentificarea clienților, conține metode pentru logare, delogare, refresh token acces, adăugare/ștergere admin
- *Client Service*
 - conține metode pentru preluarea listei de clienți, adăugare, modificare și ștergere
- *Order Service*
 - conține metode pentru adăugarea, preluarea listei de comenzi, actualizarea comenzilor sau a statusului lor și ștergerea comenzilor
- *PrefabItem Service*
 - conține metode pentru adăugarea, actualizarea, ștergerea produselor din aplicație, alături de metode pentru preluarea de istoric stoc, adăugarea de istoric stoc, categorie produs și preluarea produselor per categorie/preluarea categoriilor
- *State Service*
 - conține toate datele distribuite în aplicație
 - având dimensiuni relativ mici și nevoia ca datele să fie împărțite între mai multe componente, tot state-ul aplicației este memorat într-un singur serviciu
 - acesta conține constante
 - URL-ul backendului aplicației
 - denumirile interne pentru produsele default
 - dar și variabile utilizate de componente
 - statusul logării utilizatorului
 - statusul de admin al utilizatorului
 - tema curentă a aplicației
 - date despre comanda curentă
 - date despre produsul în proces de adăugare
 - date despre produsul vizualizat în mod curent
- *Stats Service*
 - conține metode pentru preluarea de statistici ale pentru aplicație
- *Theme Service*
 - conține o metodă pentru modificarea temei aplicației, folosindu-se de un `@Inject` pentru a putea modifica href-ul elementului `<html>`

4.3.14. Modele

Pentru fiecare tip de obiect de pe partea de back-end care se dorește a fi afișat pe front-end a fost creat câte un model care să fie echivalent cu field-urile din baza de date a server-ului. Astfel, la primirea unui răspuns HTTP de la server, datele sunt convertite automat în tipurile definite de modelul fiecărui obiect.

Acest lucru permite asignarea de tipuri TypeScript pentru răspunsurile de la back-end și oferă dezvoltatorului posibilitatea de a primi sugestii de completare automată de la mediul de dezvoltare (IDE) utilizat, după cum se poate observa în imaginea de mai jos.



```
.subscribe((prefabItems: PrefabItem[]) => {
  this.prefabItems = prefabItems.filter(
    (item: PrefabItem) => !this.state.INTERNAL_NAMES.includes(item.),
  );
});

showEdit(item: PrefabItem) {
  this.clonedItems[item.id] = { ...item };
}
```

The dropdown menu shows the following suggestions:

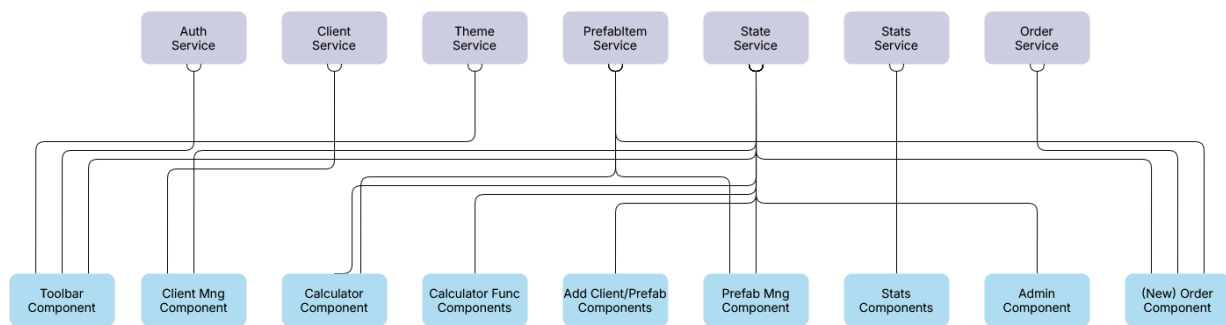
- category?
- cost?
- description?
- dimensions?
- edit_history?
- id

| | |
|--|--|
| <pre>for prefab in prefabs: category = Category.objects.filter(items__id=prefab.id).first() return_value.append({ "id": prefab.id, "name": prefab.name, "description": prefab.description, "thumbnail": prefab.thumbnail, "internalname": prefab.internalname, "instock": prefab.instock, "stock": prefab.stock, "price": prefab.price, "cost": prefab.cost, "category": { "label": category.name if category else "Necategorizat", "value": category.id if category else 0, }, "product_code": prefab.product_code, "dimensions": getattr(prefab, "dimensions", {}), })</pre> | <pre>export interface PrefabItem { id: number; name?: string; description?: string; thumbnail?: string; internalname?: string; instock?: number; stock?: number; price?: number; cost?: number; category?: object; product_code?: string; dimensions?: any; edit_history?: any; quantity?: number; }</pre> |
|--|--|

Obiectul JSON returnat de backend

Modelul de pe front-end

Putem observa în imaginile de mai sus corespondența dintre field-urile obiectului returnat de backend și modelul de pe front-end.



Relația dintre componente și servicii

În figura de mai sus se observă legătura dintre componentele principale ale aplicației și serviciile aferente. Putem observa că fiecare serviciu are măcar o componentă asociată; majoritatea serviciilor utilizează state-ul în mod direct, fără a fi necesare implementări în componentă.

5. Concluzii

Lucrarea de față a demonstrat necesitatea și beneficiile unei aplicații dedicate optimizării proceselor de producție, gestiunii stocurilor și managementului comenzilor în industria prefabricatelor din beton. Implementarea tehnologiilor moderne precum Django pentru backend și Angular pentru frontend a permis realizarea unei soluții scalabile, sigure și ușor de utilizat.

Printre principalele realizări ale proiectului se numără:

- Integrarea unui sistem eficient de gestiune a stocurilor, reducând erorile umane și optimizând costurile de operare.
- Automatizarea verificării comenzilor și gestionarea acestora într-un mod transparent și centralizat.
- Utilizarea unei interfețe moderne, bazate pe principiile **Uniform Design**, pentru o experiență de utilizare intuitivă și eficientă.
- Securizarea datelor prin autentificare bazată pe token-uri JWT și implementarea unor măsuri de protecție împotriva atacurilor comune.

În urma implementării și testării aplicației, s-a constatat că aceasta aduce o îmbunătățire semnificativă față de soluțiile existente, reducând timpul necesar procesării comenzilor și îmbunătățind comunicarea între departamentele implicate.

6. Direcții de viitor

Pentru a îmbunătăți și extinde aplicația, următoarele direcții de dezvoltare pot fi luate în considerare:

1. Integrarea unui modul de predicție a stocurilor

- Utilizarea algoritmilor de **machine learning** pentru a prezice necesarul de materiale în funcție de comenzile anterioare și tendințele pieței.
- Implementarea unui sistem de alertare automată pentru reîncărcarea stocurilor.

2. Extinderea funcționalităților pentru clienți

- Dezvoltarea unui portal web dedicat clienților, unde aceștia pot plasa comenzi și urmări statusul acestora în timp real.
- Integrarea cu servicii de plată online pentru a permite procesarea automată a comenzilor.

3. Optimizarea performanței aplicației

- Implementarea unui sistem de caching pentru reducerea timpului de încărcare a interfeței.
- Optimizarea interogărilor bazei de date pentru o performanță mai bună în cazul unui număr mare de utilizatori.

4. Dezvoltarea unei aplicații mobile

- Crearea unei aplicații native pentru Android și iOS, oferind acces rapid la datele din sistem pentru administratorii și operatorii de producție.
- Implementarea de notificări push pentru actualizări despre stocuri și comenzi.

5. Integrarea cu sisteme ERP existente

- Posibilitatea de a conecta aplicația cu soluții precum **SAP**, **Oracle ERP** sau **Microsoft Dynamics** pentru o gestionare mai complexă a resurselor.

Prin aplicarea acestor îmbunătățiri, aplicația ar putea deveni o soluție completă pentru digitalizarea și automatizarea proceselor din industria prefabricatelor din beton, oferind un avantaj competitiv semnificativ pentru utilizatorii săi.

Bibliografie

1. <https://docs.python.org/>
2. <https://docs.djangoproject.com/>
3. <https://angular.io/docs>
4. <https://www.electronjs.org/docs>
5. <https://www.postgresql.org/docs/>
6. <https://www.django-rest-framework.org/>
7. <https://pyjwt.readthedocs.io/>
8. <https://www.primefaces.org/primeng/>
9. <https://rxjs.dev/>

