

# InvoicePro

Aplicație de gestiune a facturilor unei companii

Studenti: *Hosu Razvan, Nistor Dorin, Pop Alex, Pop Cristian, Vint Alexandru*

Ianuarie 2025

# Cuprins

<b>1</b>	<b>Introducere</b>	<b>2</b>
<b>2</b>	<b>Analiza și viziunea proiectului</b>	<b>3</b>
2.1	Contextul Aplicației . . . . .	3
2.2	Produsul final dorit . . . . .	3
<b>3</b>	<b>Aspecte teoretice</b>	<b>4</b>
<b>4</b>	<b>Implementarea Aplicației</b>	<b>6</b>
4.1	Modulul de Login . . . . .	6
4.2	Modulul de utilizator . . . . .	6
4.3	Modulul de client . . . . .	8
4.4	Modulul de servicii . . . . .	9
4.5	Modulul de factură . . . . .	10
4.6	Evenimente . . . . .	12
<b>5</b>	<b>Testare și Validare</b>	<b>13</b>
5.1	Testări manuale . . . . .	13
5.2	Unit Testing . . . . .	13
<b>6</b>	<b>Rezultate</b>	<b>15</b>
<b>7</b>	<b>Concluzii</b>	<b>17</b>
<b>8</b>	<b>Referințe</b>	<b>18</b>

# 1 Introducere

**InvoicePro** este o aplicație Web concepută pentru a ajuta companiile în a-și gestiona facturile emise către clienți, respectiv de a încărca toate aceste facturi în *Spațiul Privat Virtual* al ANAF. Aplicația este destinată companiilor românești, pentru a le ușura munca când vine vorba de a încărca toate facturile în SPV.

Conform legislației românești în vigoare (*OUG. 120/4 oct. 2021*), orice factură trebuie transmisă în sistemul național privind factura electronică *RO e-Factură*. Aceste facturi trebuie să fie în format XML, lucru ce ar pune în dificultate multe persoane responsabile de facturări sau s-ar pierde mult timp cu generarea unei facturi în format XML. Astfel că, aplicația va putea genera o factură XML sau va putea încărca direct în sistemul național acest fișier.

Cu o interfață intuitivă și ușor de navigat, *InvoicePro* se adresează atât persoanelor care au mai lucrat cu sisteme de facturare, cât și acelor persoane care se întâlnesc pentru prima dată cu un sistem de gestiune a facturilor. Astfel, companiile vor putea emite și să gestioneze facturile într-un mod foarte simplu și intuitiv, optimizându-și viteza de lucru. În plus, aplicația are și un sistem de gestiune a clienților, respectiv de gestiune a serviciilor - pe care se pot aplica diverse cote de TVA (ca de pildă 9%, 12%, 19%, etc).

Aplicația este realizată în PHP 8, folosind framework-ul Laravel. Am ales acest limbaj și framework, deoarece este unul din cele mai populare limbaje, fiind ușor de învățat și folosit, iar framework-ul implementează cu succes arhitectura Model-View-Controller. Pentru partea de Frontend, am ales să lucrăm cu Tailwind pentru CSS, respectiv cu Vue.JS pentru crearea componentelor dinamice.

Am ales acest proiect într-un mod aleatoriu, luând toate titlurile și punându-le pe *random.org*. La final, ne-a rezultat acest proiect, și anume *aplicație de gestiune a facturilor*. După ce am văzut despre ce este vorba în acest proiect, am mers mai departe cu această temă.

## 2 Analiza și viziunea proiectului

### 2.1 Contextul Aplicației

Așa cum este precizat și în introducere, aplicația vine ca o soluție pentru companiile românești, având în vedere faptul că legislația română obligă firmele să încarce toate facturile emise în *Spațiul Privat Virtual al ANAF*. Astfel că procesul de facturare este automatizat, deoarece firmele nu trebuie să își genereze singure fișierul XML, respectiv să-și piardă timpul cu generarea unei facturi într-o aplicație, iar într-o altă aplicație să genereze XML-ul, în final trebuind să încarce în a treia aplicație fișierul rezultat.

Toate funcționalitățile aplicației sunt în conformitate cu documentul *Cerințe proiect "Gestiunea Facturilor"* din 17 octombrie 2024, respectiv analiza amplă a acestora a fost făcută în documentul *Software Requirements Specification - Gestiunea Facturilor*. Cerințele sunt preluate de la clientul nostru, aplicația finală fiind folosită intern, atât de către angajații care se ocupă de vânzări, cât și de administratorii firmei.

### 2.2 Produsul final dorit

Aplicația trebuie să conțină un modul de gestiune facturi - care permite încărcarea acestora în SPV, un modul de gestiune a serviciilor - care are ca rol strict stocarea informațiilor despre serviciile folosite și nu ține loc de un posibil modul de gestiune a stocurilor, un modul de gestiune a clienților - care are ca scop stocarea unor informații despre clienți, respectiv un modul de administrare utilizatori.

În final, aplicația trebuie să fie capabilă să genereze facturi, să afișeze diverse statistici, să poată stocazeze informații despre serviciile pe care le prestează compania, să poată gestiona clienții, respectiv utilizatorii să poată accesa aceste module în funcție de permisiunile pe care le au. Produsul final pe care dorim să-l obținem este descris mai pe larg în documentul *Software Requirements Specification - Gestiunea Facturilor*.

### 3 Aspecte teoretice

Când vine vorba de dezvoltarea unei aplicații, este important de luat în calcul modelul arhitectural pe care-l vom folosi. Unul dintre modelele arhitecturale foarte populare este *Model-View-Controller* (MVC). Această arhitectură împarte aplicația în 3 componente: *Model*, care are rolul de a gestiona datele, *View*, care reprezintă partea de frontend a aplicației, respectiv *Controller*, care face legătura dintre Model și View [1]. Graficul de mai jos arată cum se manifestă arhitectura MVC.

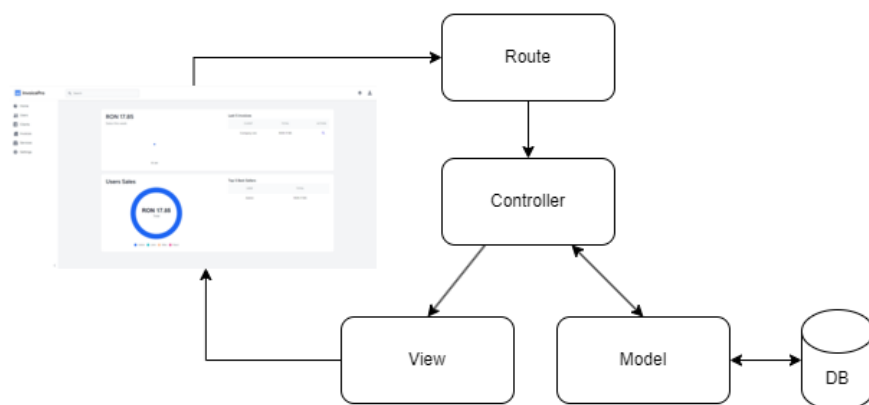


Figura 1: Grafic pentru arhitectura MVC

Laravel este un framework care implementează total MVC-ul, având o mulțime de caracteristici ce accelerează procesul de dezvoltare web. Astfel că Laravel automatizează procesele care nu sunt business logic, permițând programatorilor să se focuseze pe partea de dezvoltare a componentelor aplicației [2].

Când vine vorba de dezvoltarea unei aplicații web, este important ca timpul de răspuns sau de încărcare a unei pagini să fie unul extrem de rapid. Astfel că e nevoie de optimizări. O metodă de optimizare este de a cache-ui datele sau fișierele, astfel încât la încărcarea paginii, datele să nu fie preluate de pe server, ci local din cache. Acest lucru permite ca încărcarea unei pagini să se facă într-un mod mult mai rapid. O altă metodă de optimizare este de a minifica CSS-urile și JavaScript-urile. Acest lucru face ca fișierul să aibă o dimensiune mai mică, astfel că la încărcarea unei pagini web, descărcarea acestor fișiere se va face într-un mod extrem de rapid [3].

Pe lângă descărcarea diverselor fișiere (CSS, JavaScript, imagini, etc), un alt lucru care afectează performanța unei aplicații web este citirea din baza de date. Astfel, Laravel oferă posibilitatea ca datele să fie cache-uite într-o bază de date, fișier, Memcached sau Redis [4].

Înainte de a discuta despre aplicația de facturare, este important să clarificăm ce este o factură electronică. E-Invoice - fiind unul din cele mai folosite documente electronice - este unul din cele mai importante documente pentru o afacere. Acesta utilizează limbajul XML, în lume existând sute de standarde de implementare. Dintre

acestea, Uniunea Europeană a adoptat doar câteva standarde, precum UBL, sau XML-GS1 [5]. Deoarece fiecare țară își reglementează un standard, acesta este principalul motiv pentru care există sute de standarde la nivel mondial, fără a fi adoptat un standard comun la nivel global [6].

Când vine vorba de sisteme de facturare, cele mai comune module implementate sunt modulul de autentificare, modulul de facturare - mai precis generare factură și vizualizare, modul de servicii sau produse, respectiv modul de utilizatori [7]. Recent, însă, au început să apară pe piață și module de e-Invoice, ce permit generarea unor facturi în format XML [8].

În plus, sistemele de facturare trebuie să conțină un modul de setări ale companiei, unde se trec datele de identificare fiscală a companiei care emite facturile, precum numele companiei, codurile de identificare, adrese, cote TVA și așa mai departe. Pe lângă acest modul, sistemele de facturare pot conține și diverse statistici cu privire la facturile emise [9].

Când vine vorba de sisteme ce sunt conectate la diferite API-uri, unele dintre ele folosesc *OAuth*. OAuth 2.0 este o metodă de autentificare sigură, ce validează credențialele și returnează la final un token cu care se pot accesa respectivele API-uri. Pentru a primi un token, pentru început e nevoie de un *authorization code*, ce se obține prin validarea unor credențiale. După obținerea acestui cod, se trece la pasul următor care reprezintă obținerea tokenului folosind codul primit. Cu acest cod, se va face request către server, iar dacă codul este valid, atunci se va returna tokenul. Acest lucru deja se poate face ușor folosind Laravel, pentru a reduce timpul de dezvoltare [10].

## 4 Implementarea Aplicației

### 4.1 Modulul de Login

Modulul de autentificare include o pagină de logare unde utilizatorii își vor putea introduce datele de acces, mai exact emailul și parola pentru a se autentifica. Accesul se realizează pe baza conturilor deja create de către administrator, deoarece utilizatorii nu își vor putea crea conturi.

Prin introducerea unui e-mail și a unei parole pentru autentificare, aceste credențiale vor fi verificate cu ajutorul controllerului de autentificare. În cazul în care credențialele sunt valide, utilizatorul va fi considerat logat și va fi trimis pe pagina principală. În caz contrar, se va returna un mesaj de eroare.

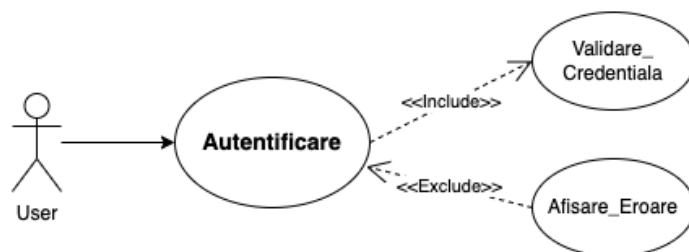


Figura 2: Use Case pentru Autentificare

Dacă parola a fost uitată, aplicația include un controller pentru resetarea acesteia. Procesul presupune validarea identității utilizatorului prin e-mail și accesul acestuia la adresa respectivă. Va fi trimis un e-mail cu instrucțiuni pentru resetarea parolei, iar autentificarea în aplicație va deveni posibilă după finalizarea acestui proces.

### 4.2 Modulul de utilizator

Modulul de utilizatori are următoarele componente: adăugare/editare utilizatori, listare utilizatori, detalii utilizator, ștergere utilizator. Pentru fiecare componentă există câte un controller separat, astfel:

Componentă	Controller
Listare utilizatori	Users\UsersController
Detalii utilizator	Users\UserDetailsController
Formular adăugare/editare utilizator	Users\UsersFormController
Ștergere utilizator	Users\DeleteUserController

Tabela 1: Corespondența componentă-controller pentru modulul de utilizatori

Listarea utilizatorilor se face selectând toți userii care nu au fost șterși, mai precis în baza de date au coloana *deleted\_at* setată ca *NULL*. De asemenea, datele sunt paginate, astfel că pe partea de View vor apărea doar câte 10 useri, în funcție de pagina selectată.



Figura 3: Diagrama de clasă pentru fiecare controller din modulul utilizatori

Formularul pentru adăugare sau editare useri are următoarele categorii: detalii generale și permisiuni. Detaliile generale pentru un utilizator sunt următoarele: numele, email-ul și numărul de telefon. Administratorii nu vor putea edita parolele userilor, din motive de siguranță. Permisunile utilizatorilor sunt de tipul *[componenta]-[modul]*. De pildă, pentru modulul de useri, permisiunile sunt următoarele: *vizualizare useri*, *formular useri*, *ștergere useri*. Același tipar se respectă pentru toate modulele. Totodată, este posibilă bifarea unei opțiuni "setează ca admin", care va anula posibilitatea de a selecta permisiuni, însă va considera că utilizatorul va fi admin.

La trimiterea formularului (submit), datele introduse vor fi validate. În cazul în care există probleme, ca de pildă email-ul este deja folosit sau lipsește numele, atunci se va reveni înapoi la formular și se vor afișa erorile apărute. În cazul în care toate câmpurile sunt valide. Toate datele din POST vor fi procesate, iar utilizatorul va fi adăugat sau editat, în funcție de tipul de acțiune. Totodată, dacă acțiunea este de adăugare, atunci utilizatorului i se va genera o parolă aleatorie.

La detaliul utilizatorului vor fi afișate informațiile relevante, precum numele, email-ul, permisiunile, respectiv dacă este online sau nu. Ultima componentă a acestui modul este ștergerea utilizatorului, care nu este o ștergere propriu-zisă, ci este vorba de *softdelete*, adică utilizatorului i se va seta în baza de date pe coloana *deleted\_at* data și ora curentă.

În mod normal, această secțiune ar fi destinată administratorilor, însă pot exista și angajați care să se ocupe cu adăugarea sau modificarea utilizatorilor. Figura de mai jos reprezintă Use Case-ul pentru acest modul.

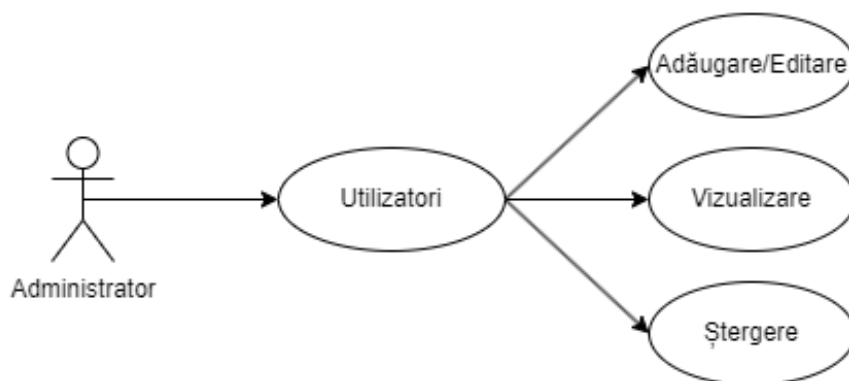


Figura 4: Use Case pentru modulul de utilizatori



### 4.3 Modulul de client

Modulul de clienți este responsabil pentru informațiile legate de clienții aplicației, oferind componente precum adăugarea sau modificarea clienților, vizualizarea și ștergerea acestora. Acesta permite utilizatorilor să stocheze și să actualizeze datele clienților, aceste date fiind folosite în procesul de facturare.

Componentă	Controller
Listare clienți	Clients\ClientsController
Detalii client	Clients\ClientDetailsController
Formular adăugare/editare clienți	Clients\ClientsFormController
Ștergere client	Clients\DeleteClientController

Tabela 2: Corespondența componentă-controller pentru modulul de clienți



Figura 5: Diagrama de clasă pentru fiecare controller din modulul de clienți

Listarea clienților se face selectând toți clienții care nu au fost șterși. De asemenea, datele sunt paginate, astfel că pe partea de View vor apărea doar câte 10 clienți, în funcție de pagina selectată.

Formularul de adăugare sau modificare clienți este împărțit pe 2 categorii: detalii generale și persoane de contact. Detaliile generale conțin următoarele câmpuri: nume, cod de identificare, tip client - care poate fi persoana sau firma, adresa, țară, județ, localitate. Categoria de persoane de contact se ocupă de afișarea persoanelor de contact curente - ce pot fi editate sau șterse, respectiv de adăugarea unor noi contacte. Câmpurile pentru o persoană de contact sunt următoarele: numele, prenumele, titlul (funcția sau ocupația în firmă), email și număr de telefon.

La trimiterea formularului (submit), datele introduse vor fi validate. În cazul în care există probleme, ca de pildă lipsește un câmp, atunci se va reveni înapoi la formular și se vor afișa erorile apărute. În cazul în care toate câmpurile sunt valide, se va adăuga noul client (dacă acțiunea este de adăugare) sau se va edita. La editare, totodată, se vor șterge persoanele de contact în totalitate, iar mai apoi se vor adăuga în baza de date noile persoane de contact.

La detaliu client vor fi afișate informațiile generale despre client, persoanele de contact, respectiv facturile care s-au emis către acesta. Ultima componentă a acestui modul este ștergerea clientului, fiind de fapt softdelete. Clientul nu este șters propriu-zis, ci doar i se setează în coloana *deleted\_at* data și ora curentă.

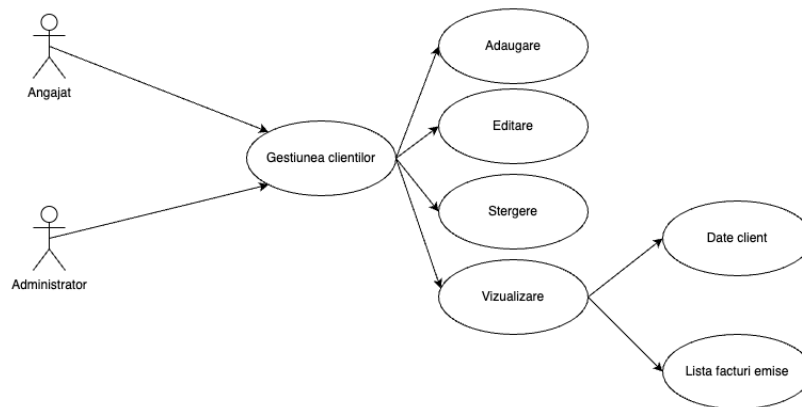


Figura 6: Use Case pentru modulul de clienți

## 4.4 Modulul de servicii

Modulul de servicii stochează serviciile pe care le prestează o companie. Este echivalentul modulului de inventar (sau produse), diferența fiind că în cazul de față, compania pentru care se face această platformă prestează servicii și nu vinde produse. Componentele acestui modul sunt adăugarea/editarea unui serviciu, vizualizarea unui serviciu, listarea serviciilor și ștergerea unui serviciu.

Componentă	Controller
Listare servicii	Services\ServicesController
Detalii serviciu	Services\ServiceDetailsController
Formular adăugare/editare servicii	Services\ServicesFormController
Ștergere serviciu	Services\DeleteServiceController

Tabela 3: Corespondența componentă-controller pentru modulul de servicii

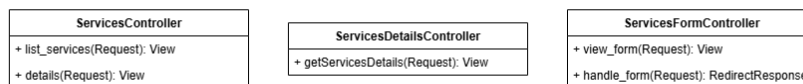


Figura 7: Diagrama de clasă pentru fiecare controller din modulul de servicii

Listarea serviciilor se face selectând toate serviciile care nu au fost șterse. De asemenea, datele sunt paginate, astfel că pe partea de View vor apărea doar câte 10 clienți, în funcție de pagina selectată.

Formularul de adăugare sau modificare servicii conține următoarele câmpuri: nume, preț, TVA, moneda, descriere. La trimitere, datele se validează. Dacă în timpul validării apar erori, ca de pildă numele nu a fost setat, atunci se va reveni pe pagina cu formularul

și se vor afișa erorile de validare. Dacă validarea a trecut cu succes, atunci noul serviciu se adaugă în baza de date, dacă acțiunea este una de adăugare, sau serviciul va fi editat, în caz contrar.

La detaliu serviciu vor fi afișate informațiile despre serviciu. Ultima componentă a acestui modul este ștergerea serviciului, fiind de fapt softdelete. Serviciul nu este șters propriu-zis, ci doar i se setează în coloana *deleted\_at* data și ora curentă.

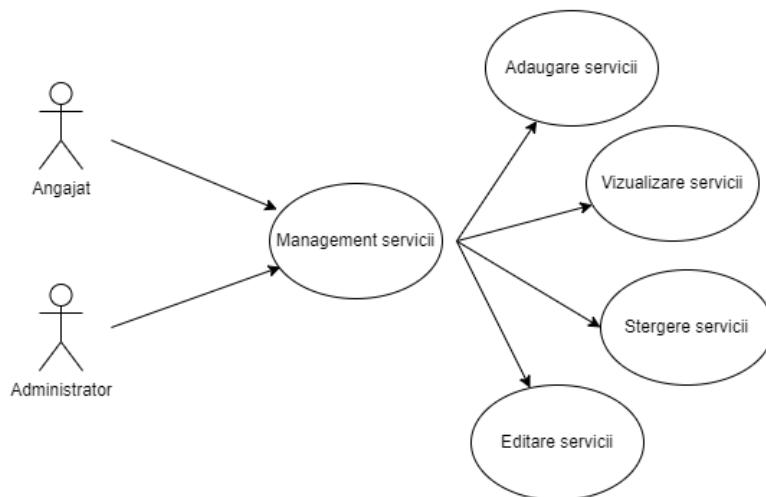


Figura 8: Use Case pentru modulul de servicii

## 4.5 Modulul de factură

Modulul de facturi este destinat să faciliteze procesul de creare, gestionare și trimitere a facturilor. Acesta ajută la automatizarea și simplificarea activităților administrative legate de facturare, reducând riscul major de erori și economisind semnificativ timp. Totodata, acest modul este integrat cu platforma SPV al ANAF. Componentele acestui modul sunt următoarele: listare facturi, detaliu factură, printare factură, adăugare sau editare factură și exportarea facturii în SPV.

Componentă	Controller
Listare facturi	Invoices\InvoicesController
Detalii factură	Invoices\InvoiceDetailsController
Formular adăugare/editare factură	Invoices\InvoicesFormController
Ștergere factură	Invoices\DeleteInvoiceController
Printare factură	Invoices\InvoicePrintController
Exportare factură în SPV	Invoices\ExportSpvController

Tabela 4: Corespondența componentă-controller pentru modulul de facturi



Figura 9: Diagrama de clasă pentru pentru fiecare controller din modulul de factură

Listarea și ștergerea unei facturi funcționează precum listarea și ștergerea în celelalte module. În schimb, detaliu factură este diferit. Aici va fi randată factura, ce va conține următoarele: informații despre client, informații despre compania care emite factura, respectiv lista cu serviciile pe care le conține factura, cu preț, cantitate, cotă tva și preț final.

Formularul de adăugare sau editare factură are următoarele câmpuri: client, monedă, respectiv o listă de servicii. Aici se pot adăuga, edita sau șterge diverse servicii. În cazul în care se adaugă sau se editează un serviciu, se va deschide un modal, ce conține următoarele câmpuri: numele serviciului, preț, tva, monedă, cantitate. În cazul în care serviciul deja există în modulul de servicii, toate câmpurile vor fi completate automat, cu excepția câmpului de cantitate.

La trimiterea formularului, în cazul în care câmpurile din POST au trecut de validare, datele prelucra, iar mai apoi se va adăuga sau modifica factura în baza de date, în funcție de tipul de acțiune, și se va apela un eveniment de calculare a totalului facturii.

Totodată, există posibilitatea ca factura să fie printată sau salvată în format PDF. În plus, există posibilitatea ca factura să fie încărcată în Spațiul Privat Virtual. La apăsarea butonului de exportare în SPV din detaliu factură, se va apela ruta de *Invoices\ExportSpvController*, care va seta starea SPV în *pending*, iar apoi va apela un eveniment care va avea rolul de a trimite factura în SPV.

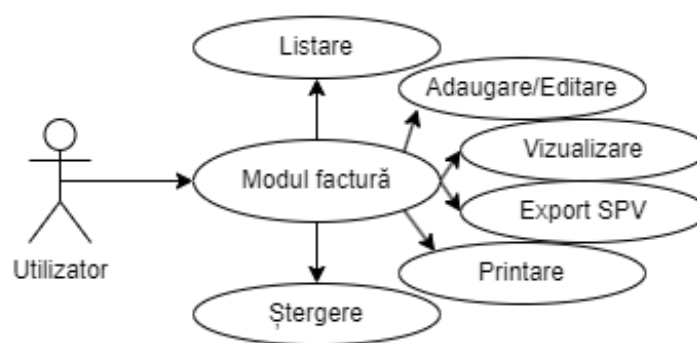


Figura 10: Use Case pentru modulul de factură

## 4.6 Evenimente

### 4.6.1 Eveniment de adăugare și editare factură

Pentru adăugare și editare factură, există două evenimente, *Events\AddedInvoice* și *Events\AddedInvoice*. Acestor evenimente li se atașează un listener, numit *Listeners \ CalculateInvoiceValue*.

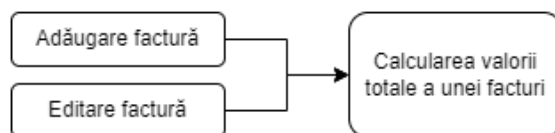


Figura 11: Diagramă reprezentând evenimentele de adăugare și editare factură

Listenerul *CalculateInvoiceValue* are rolul de a calcula valoarea totală a unei facturi, convertind toate prețurile serviciilor (care pot să fie în diverse monede) la moneda setată pentru facturare. Această conversie se face la cursul curent, printr-un API public.

### 4.6.2 Eveniment de încărcare a facturii în SPV

Acestui eveniment, *Events\ExportInvoiceToSpv*, i se atașează un listener, numit *Listeners\ExportInvoiceToSpvListener*. Acest eveniment, respectiv listener, au rolul de a încărca o factură în SPV.

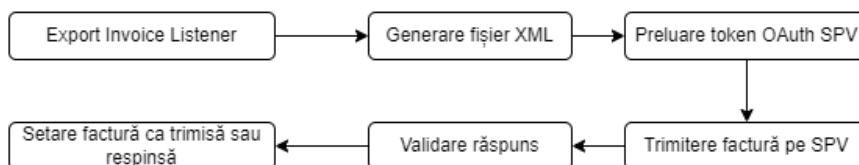


Figura 12: Diagramă reprezentând procesul de încărcare a facturii în SPV

După cum se poate vedea și în figura de mai sus, în primă etapă se generează fișierul XML al facturii. În a doua etapă, se preia token-ul OAuth din SPV, folosind ID-ul de client, respectiv Client Secret-ul. Dacă se obține un token valid, atunci se trece la etapa următoare, și anume trimiterea facturii în SPV. Se așteaptă un răspuns, iar dacă răspunsul este *ok*, atunci se setează starea ca fiind trimisă/acceptată. În caz contrar, starea facturii va fi setată ca respinsă. Totodată, dacă autentificarea în SPV prin OAuth a eșuat, atunci starea facturii va fi setată ca fiind respinsă.

## 5 Testare și Validare

### 5.1 Testări manuale

Pe partea de View, toate modulele și componentele acestora au fost testate manual. Printre testele făcute, se numără și următoarele:

- Vizibilitatea modulelor în funcție de permisiunile pe care le au utilizatorii;
- Formularele returnează erorile, în cazul în care datele introduse sunt invalide;
- Ștergerea înregistrărilor se face prin metoda `softdelete` și nu sunt șterse complet din baza de date;
- Detaliile pentru fiecare modul sunt valide;
- Listarea înregistrărilor este corectă;
- Componentele Vue.JS nu provoacă erori;
- Scripturile JavaScript pentru diverse zone nu provoacă erori;

### 5.2 Unit Testing

#### 5.2.1 Testarea funcționalității de autentificare

Rolul acestui test este de a verifica, dacă logarea funcționează, așadar se testează următoarele:

- ruta de `/login` returnează status 200;
- userul se poate loga cu credențiale valide;
- formularul gestionează parolele greșite;
- utilizatorul se poate deloga;

#### 5.2.2 Testarea modului de clienți

Rolul acestui test este de a verifica, dacă modulul de clienți funcționează așa cum trebuie, așadar se vor testa următoarele:

- userii nelogati sunt redirecționați către ruta de `/login`;
- userul este redirectionat pe `clients/index`, în cazul în care se accesează detaliile unui client invalid;
- userul poate vedea detaliile clientului pentru ID valid;

### 5.2.3 Testarea funcționalității clasei CurrencyConverter

Rolul acestui test este de a verifica, dacă clasa CurrencyConverter returnează valori corecte, respectiv Caching-ul funcționează. Așadar, se vor testa următoarele:

- dacă preia corect conversia din cache;
- în cazul în care cursul nu este cache-uit, să preia din API;
- să returneze 0, dacă API-ul e picat;
- să returneze 0, dacă moneda nu este suportată de API;

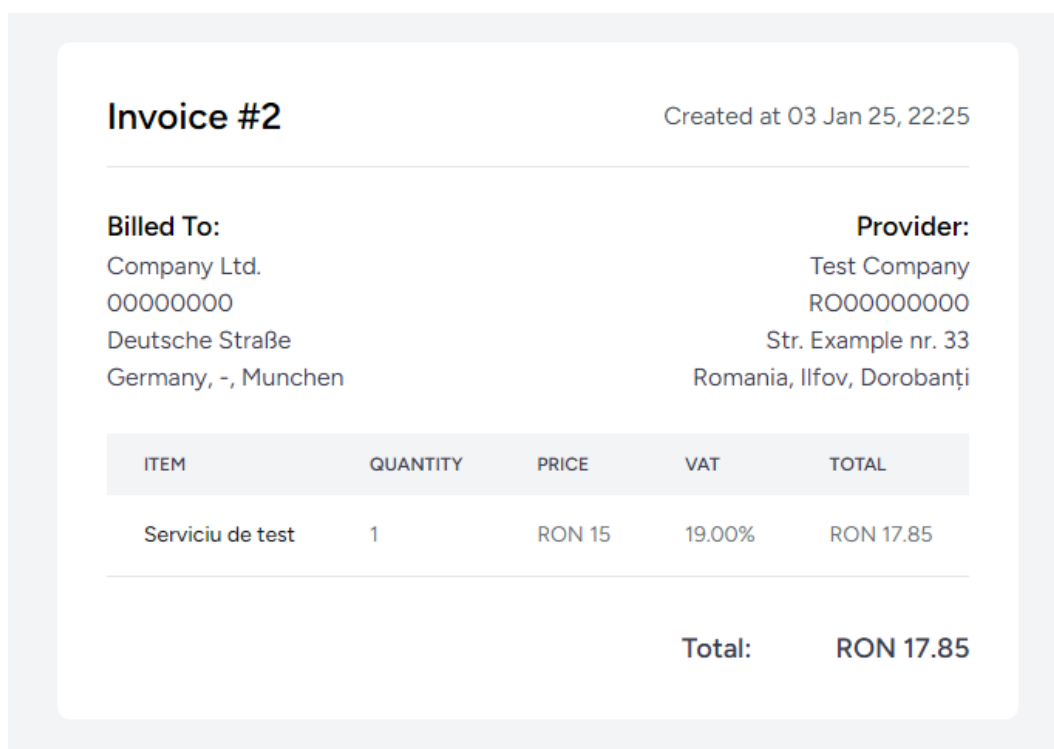
### 5.2.4 Testarea funcționalității clasei Settings

Rolul acestui test este de a verifica, dacă clasa Settings returnează valori corecte, respectiv dacă Caching-ul funcționează.

## 6 Rezultate

Proiectul final este capabil să genereze facturi, să gestioneze clienții, serviciile, respectiv utilizatorii, dar și să permită interacțiunea utilizatorului cu modulele aplicației în funcție de permisiunile pe care le are.

Cea mai importantă parte a acestei aplicații este modulul de facturi. În figura de mai jos se poate vedea cum arată o factură generată de aplicația noastră. Factura conține datele de facturare ale clientului, datele de identificare ale companiei care emite factura, serviciile cu numele serviciilor, cantitățile, prețul per unitate, tva și prețul final, respectiv totalul facturii.



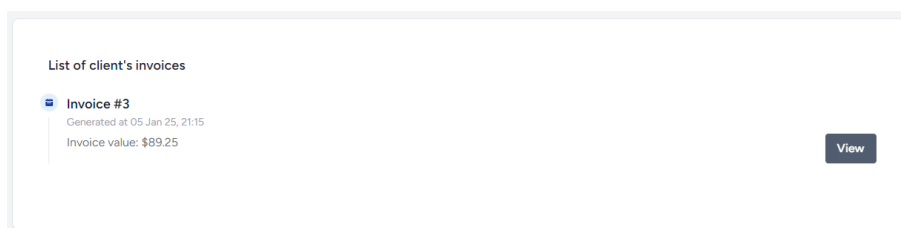
The screenshot displays a web interface for an invoice. At the top left, it says 'Invoice #2'. At the top right, it says 'Created at 03 Jan 25, 22:25'. Below this, there are two columns of information: 'Billed To:' and 'Provider:'. The 'Billed To:' column contains 'Company Ltd.', '00000000', 'Deutsche Straße', and 'Germany, -, Munchen'. The 'Provider:' column contains 'Test Company', 'RO00000000', 'Str. Example nr. 33', and 'Romania, Ilfov, Dorobanți'. Below this, there is a table with 5 columns: 'ITEM', 'QUANTITY', 'PRICE', 'VAT', and 'TOTAL'. The table has one row with the following data: 'Serviciu de test', '1', 'RON 15', '19.00%', and 'RON 17.85'. At the bottom right, there is a 'Total:' label followed by 'RON 17.85'.

ITEM	QUANTITY	PRICE	VAT	TOTAL
Serviciu de test	1	RON 15	19.00%	RON 17.85

Total: RON 17.85

Figura 13: Factură generată din aplicație

Conform cerințelor, aplicația gestionează clienții cărora compania emite facturi. Una dintre cele mai importante lucruri este că utilizatorul poate vedea atât datele de identificare ale companiei, cât și persoanele de contact. În plus, utilizatorul poate vedea și facturile emise către acesta.



The screenshot shows a web interface titled 'List of client's invoices'. It contains a list of invoices. The first invoice is 'Invoice #3', generated at '05 Jan 25, 21:15', with an 'Invoice value: \$89.25'. There is a 'View' button next to the invoice details.

Invoice #	Generated at	Invoice value
Invoice #3	05 Jan 25, 21:15	\$89.25

View

Figura 14: Facturile emise către un client specific



Client Details

Edit

NAME	CLIENT TYPE	CUI
CodeStream Inc.	individual	98144466

ADDRESS

LOCATION

505 Compiler Court, Dev Center, DC 55005

Fort Pierce, Florida, United States of America

Contact

NAME	TITLE	PHONE	EMAIL
John Smith	CEO	555-255-25-25	john@codestream.com
Mike Tyson	CFO	555-303-30-30	mike@codestream.com

Figura 15: Detaliile și persoanele de contact ale clientului

## 7 Concluzii

InvoicePro a fost dezvoltată ca o soluție practică și eficientă pentru gestionarea facturilor, răspunzând nevoilor atât ale administratorilor, cât și ale angajaților. Prin intermediul acestui proiect, noi ca și grupă, dar și în mod individual, am acumulat o experiență valoroasă în proiectarea și implementarea unei aplicații din lumea reală care abordează provocările comune în implementare și dezvoltare.

În timp ce aplicația își îndeplinește bine funcțiile de bază, există câteva caracteristici care pot fi adăugate pentru a o îmbunătăți și mai mult, precum:

- Adăugarea a mai multor diagrame sau grafice pentru o perspectivă mai bună asupra tendințelor de facturare.
- Compatibilitate cu dispozitivele mobile, pentru crearea unui design receptiv pentru utilizare pe smartphone-uri.
- Suport în mai multe limbi, pentru a satisface o gamă mai largă de utilizatori.

Nu totul a decurs fără probleme, repararea erorilor neașteptate, gestionarea relațiilor cu bazele de date și perfecționarea interfeței cu utilizatorul au fost mai dificile decât ne-am imaginat. Dar aceste provocări au făcut proiectul mai interesant. Am învățat importanța persistenței și cum să abordăm problemele metodic, iar prin acest prilej mulțumim pentru această oportunitate.

## 8 Referințe

- [1] Rinaldo, Juwari, Karno Diantoro, Abdur Rohman, Anwar T. Sitorus, *Optimizing Invoice Management: A Case Study of PT. Madina Mitra Teknik's Transition to a Laravel-Based Information System*, Digitus : Journal of Computer Science Applications, 2023
- [2] Andri Sunardi, Suharjito, *MVC Architecture: A Comparative Study Between Laravel Framework and Slim Framework in Freelancer Project Monitoring System Web Based*, 4th International Conference on Computer Science and Computational Intelligence 2019 (ICCSCI), 2019
- [3] Zamah Sari, Moechammad Sarosa, Heru Nurwasito, *Concept of Designing an Optimized Pull Model View Controller Type Content Management Framework*, International Journal of Computer Applications (0975 – 8887), 2012
- [4] Sadik Khan, Aasha T. Khanam, *Study on MVC Framework for Web Development in PHP*, ISSN 2456-3307, 2023.
- [5] Zvonimir Vanjak, Vedran Mornar, Ivan Magdalenic, *Deployment of e-Invoice in Croatia*, University of Zagreb, 2008
- [6] Neven Vrcek, Ivan Magdalenic, *Methodology and Software Components for E-Business Development and Implementation: Case of Introducing E-Invoice in Public Sector and SMEs*, Journal of Cases on Information Technology, 2011
- [7] Stephen Luke Harris, *Invoice System*, National College of Ireland, 2017
- [8] Chukwudi Kingsley Williams, *Online Invoice Management System Software as a Service*, Metropolia University of Applied Sciences, 2017
- [9] Lele Liu, Bingjie Wang, Xiangjun He, Juan Wang, Yongfeng Zheng and Yongbing Yan, *Establishing an electronic invoice management platform based on information system*, Journal of Physics: Conference Series, 2004
- [10] Arthur Oliviana Zabka, Asep Id Hadiana, Herdi Ashaury, *Implementation of OAuth 2.0 based on Laravel Framework in a case of study of Client Information Management System*, Journal of Informatics and Communication Technology (JICT), 2023