PROIECT FINAL

**CURS JAVA**

**CURSANT**: *MELINTE VLAD-TUDOR*

2022

TEMA PROIECT: „APLICATIE DE GESTIUNE A DECLARATIILOR FISCALE”

Proiectul iși propune rezolvarea unei probleme pe care o poate întâmpina un contabil în rutina muncii sale, anume, sa poata gestiona și ține evidenta declarațiilor fiscale pe care le are de întocmit și depus pentru un portofoliu de firme.

Proiectul este structurat în două părți principale: modalitatea de gestiune a societăților din portofoliu ( creare entitate nouă, editare, ștergere) și partea de gestiune a declarațiilor ce sunt depuse (adaugare declarații, asociere, ștergere).

Alte aspecte legate de funcționalitate și accesibilitate ale aplicației ar fi: posibilitatea de filtrare, atât în ceea ce privește gestiunea societăților, cât și a declarațiilor; posibilitatea de căutare în baza de date dupa nume societate, cod fiscal, adresa, etc.; posibilitatea adăugării de observații în momentul înregistrarii depunerii de declarații, observații ce ar putea fi folosite ulterior pentru a ajuta utilizatorul să întocmească rapoarte sau informări pentru clienții lui.

Structura bazei de date este compusa din:

-Tabela „firme” cu următoarele câmpuri:



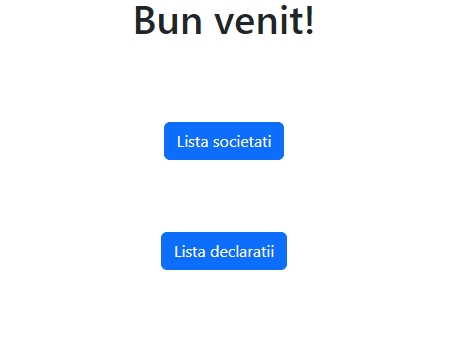
-Tabela “records” cu următoarele câmpuri:



\*Legătura între cele doua tabele se va realiza prin campurile id(firme)și id\_record(records)

Pe baza acestei relații, cu ajutorul query-urilor SQL se vor obține informațiile dorite a se afișa, respectiv va exista posibilitatea filtrarii și sortării informațiilor.

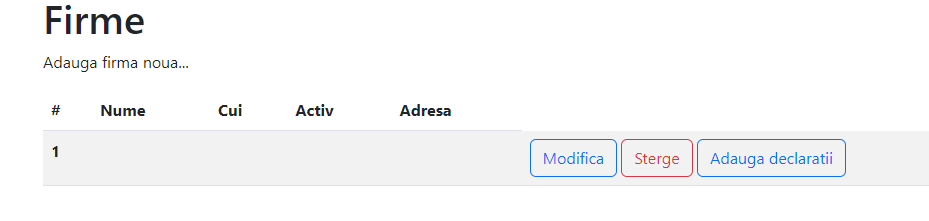
Prima pagina (Home) a aplicației afișeaza 2 butoane care conduc către cele 2 liste principale, cu informații aduse din baza de date.



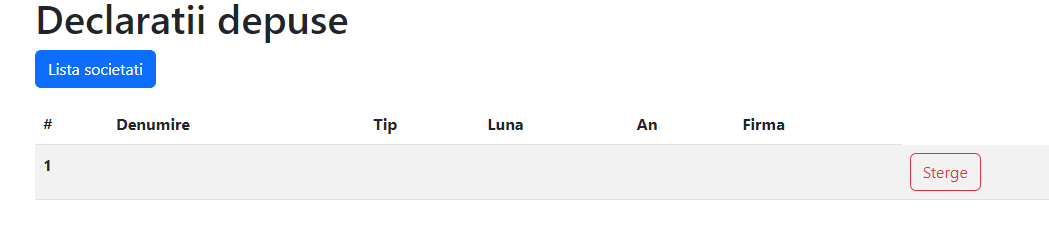
Cu ajutorul primului buton se accesează lista cu societățile înregistrate în baza de date, loc din care se poate efectua adaugarea de noi firme, modificarea informațiilor acestora, ștergerea lor și atribuirea de declarații fiscale depuse.

Al doilea buton face legatura către lista de declarații înregistrate, cu posibilitatea de a adăuga noi declarații sau de a le șterge pe cele existente, în cazul în care se dorește acest lucru.

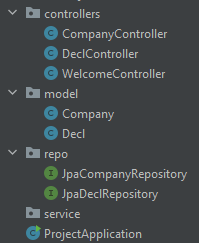
Pagina care afișează lista societăților și posibilitatea editării datelor arată în felul urmator:



Iar cea de declarații, astfel:



Structura Java a proiectului cuprinde 2 clase de tip „model” în care sunt definite entitățile ”company”(firma) și ”decl”(declaratie), 3 controllere ce se ocupa de solicitările și răspunsurile trimise și primite prin HTML, 2 interfețe de tip repository, care extind interfața Jpa Repository și clasa Main ”ProjectApplication” care se ocupă de lansarea în execuție a aplicației propriu zise.



Fișierul Company.java conține:

@Entity  
@Table(name = "firme")  
public class Company {  
 @Id  
 private UUID id;  
 @Column(name = "numeFirma")  
 private String name;  
 @Column(name = "CUI")  
 private String codF;  
 @Column(name = "Activ")  
 private boolean activity;  
 @Column(name = "Adresa")  
 private String adress;

Fișierul Decl.java cuprinde:

@Entity  
@Table(name = "records")  
public class Decl {  
 @Id  
 @NonNull  
 private UUID idRecord;  
 @Column(name = "An\_depunere")  
 private String year;  
 @Column(name = "Luna\_depunere")  
 private String month;  
 @Column(name = "Nume\_declaratie")  
 private String nameRecord;  
 @Column(name = "Tip\_declaratie")  
 private String typeRecord;  
 @Column(name = "Observatii")  
 private String obs;  
  
 @ManyToMany(fetch = FetchType.*EAGER*)  
 @JoinTable(  
 name = "company\_decl\_relation",  
 joinColumns = @JoinColumn(name="company\_id"),  
 inverseJoinColumns = @JoinColumn(name = "decl\_id")  
 )

Primul controller denumit ”HelloController” răspunde la accesarea adresei: “localhost:8080” și trimite pagina ”Home.html”

@Controller  
public class WelcomeController {  
  
 @GetMapping(path= "/")  
 public String hello1 (Model model){  
 return "home";  
 }  
}

CompanyController gestioneaza solicitările și răspunsurile pe calea ”/companies(”/”, ”/add”,”/edit/id”, ”/delete/id”)”

@GetMapping("/")  
public String listCompanies(Model model){  
 model.addAttribute("companies", jpaCompanyRepository.findAll());  
 return "companies/list";  
}  
@GetMapping("/add")  
public String addCompanyForm(Model model) {  
 return "companies/addForm";  
}  
  
@PostMapping("/add")  
public RedirectView addCompany(Model model,  
 @RequestParam("company\_name") String companyName,  
 @RequestParam("company\_codF") String companyCodF,  
 @RequestParam("company\_activity") Boolean companyActivity,  
 @RequestParam("company\_adress") String companyAdress) {  
 Company addedCompany = new Company(UUID.*randomUUID*(), companyName, companyCodF, companyActivity, companyAdress);  
  
 jpaCompanyRepository.saveAndFlush(addedCompany);  
  
 return new RedirectView("/companies/");  
}  
  
@GetMapping(path= "/edit/{id}")  
public String editCompanyForm(Model model, @PathVariable("id") UUID companyId) {  
 Optional<Company> optionalCompany = jpaCompanyRepository.findById(companyId);  
 Company company = optionalCompany.get();  
 model.addAttribute("Company", company);  
  
 return "companies/editForm";  
}  
@PostMapping("/edit/{id}")  
public RedirectView addCompany(Model model,  
 @PathVariable("id") UUID companyId,  
 @RequestParam("company\_name") String companyName,  
 @RequestParam("company\_codF") String companyCodF,  
 @RequestParam("company\_activity") Boolean companyActivity,  
 @RequestParam("company\_adress") String companyAdress) {  
 Optional<Company> company = jpaCompanyRepository.findById(companyId);  
 company.get().setName(companyName);  
 company.get().setCodF(companyCodF);  
 company.get().setActivity(companyActivity);  
 company.get().setAdress(companyAdress);  
 jpaCompanyRepository.save(company.get());  
 return new RedirectView("/companies/");  
}  
  
@GetMapping("/delete/{id}")  
public RedirectView deleteCompany(Model model, @PathVariable("id") UUID companyId) {  
 jpaCompanyRepository.deleteById(companyId);  
 return new RedirectView("/companies/");  
}

DeclController gestionează interacțiunea cu înregistrarile de declarații în baza de date:

@GetMapping("/add")  
public String addDeclForm(Model model){  
 return "declaratii/addForm";  
}  
@PostMapping("/add")  
public RedirectView adddecl(Model model,  
 @RequestParam("decl\_month") String month,  
 @RequestParam("decl\_year") String year,  
 @RequestParam("decl\_name") String name,  
 @RequestParam("decl\_type") String type,  
 @RequestParam("decl\_obs") String obs) {  
 Decl addedDecl = new Decl(UUID.*randomUUID*(), year,month ,name, type, obs);  
  
 jpaDeclRepository.saveAndFlush(addedDecl);  
  
 return new RedirectView("/companies/");  
}  
  
@GetMapping("/")  
public String listDecl(Model model){  
 model.addAttribute("declaratii", jpaDeclRepository.findAll());  
 return "declaratii/listDecl";  
}  
@GetMapping("/delete/{id}")  
public RedirectView deleteDecl(Model model, @PathVariable("id") UUID declId) {  
 jpaDeclRepository.deleteById(declId);  
 return new RedirectView("/decl/");  
}

Aplicația poate fi îmbunatățită cu un mediu de afișare filtrată a informațiilor în funcție de firma pentru care s-a facut înregistrarea de declarații, filtrata dupa luna și anul la care fac declarațiile referință, afișarea listei de declarații pentru o firma anume și într-un interval anume.

Cautarea dupa nume firma și tip de declarație sau luna/an ar mai putea reprezenta un scenariu pentru o viitoare îmbunatățire a functionalității, și totodată ar putea oferi mai multă flexibilitate în obținerea de informații pentru utilizator.

O alta îmbunătățire viitoare ar putea fi adăugarea de arhitecturii de lucru cu mai mulți utilizatori, fiecare sa dețina un cont securizat și să-și gestioneze propriile firme și înregistrări cu declarații.