ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

CONTROLE N°1
Architecture JEE

Filière :
« Ingénierie Informatique : Big Data et Cloud Computing »

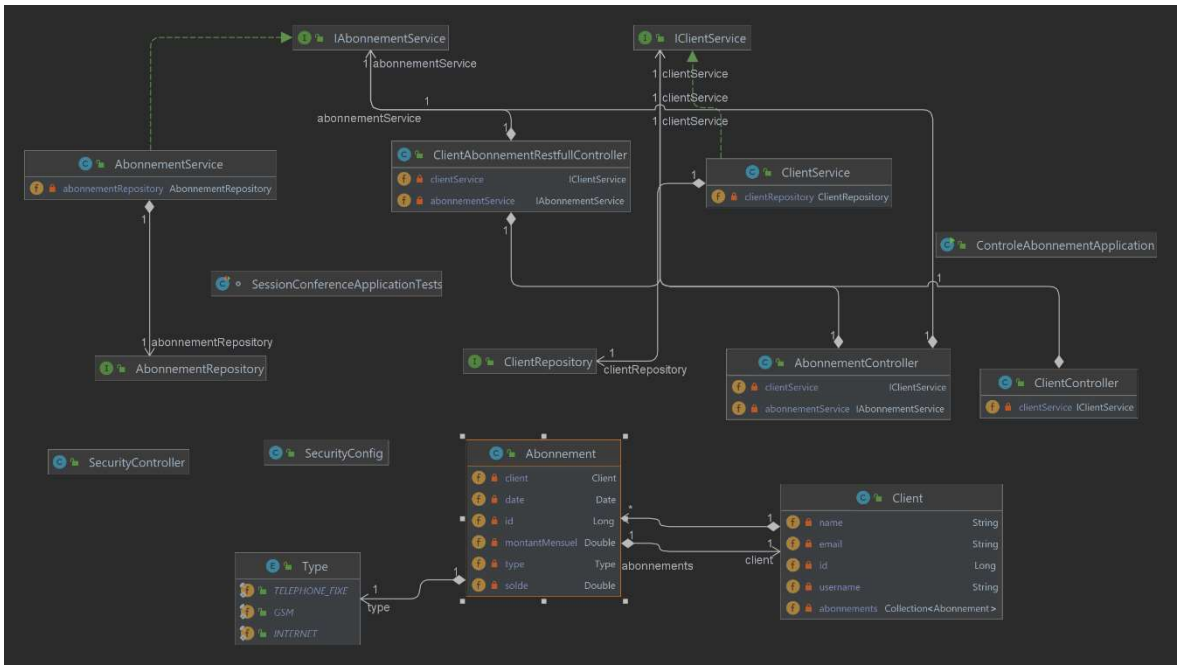
II-BDCC

Conception et développement d'une application web JEE pour la gestion des abonnements

Abderrahmane ETTOUNANI

Année Universitaire : 2022-2023

Conception



Implémentation

1- Couche DAO

- Les entités JPA

```
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Client {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    private String username;
    @OneToMany(mappedBy = "client")
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Collection<Abonnement> abonnements;
}
```

```java
@Entity
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Abonnement {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Temporal(TemporalType.DATE)
    private Date date;
    @Enumerated(EnumType.STRING)
    private Type type;
    private Double solde;
    private Double montantMensuel;
    @ManyToOne
    private Client client;
}
```

```java
public enum Type {
    GSM, INTERNET, TELEPHONE_FIXE
}
```

- Les interfaces JpaRepository

```java
public interface ClientRepository extends JpaRepository<Client, Long> {
    Page<Client> findAllByNameContains(String name, Pageable pageable);
}
```

```java
public interface AbonnementRepository extends JpaRepository<Abonnement, Long> {
    Page<Abonnement> findAllByMontantMensuel(Double montant, Pageable pageable);
}
```

- Test de la couche DAO

```java
@Bean
CommandLineRunner commandLineRunner(AbonnementRepository abonnementRepository, ClientRepository clientRepository) {
    return args -> {
        Stream.of("Ahmed", "Youssef","Abderrahmane","Youssef","Boutaina","Amin","Samir","Karima","Khadija","Fatima","Said","ELfasi","Ettounani","Mehdi","Oumaima","Hassna","Saida").forEach(
                s -> {
                    Client client = Client.builder()
                            .name(s)
                            .email(s + "@gmail.com")
                            .username(s)
                            .build();
                    clientRepository.save(client);
                }
        );
        clientRepository.findAll().forEach(
```

```
                    client -> {
                        for (int i = 0; i < 2; i++) {
                            Abonnement abonnement =
Abonnement.builder()
                                    .solde(1020.9* Math.random()+100)
                                    .client(client)
                                    .date(new Date())

.montantMensuel(10220.9*Math.random()+23)
                                    .type(Math.random() > 0.5 ?
Type.GSM : Type.TELEPHONE_FIXE)
                                    .build();
                            abonnementRepository.save(abonnement);
                        }
                    }
            );


        };
    }
```

2- Couche Web

- Gestion des clients

```
@Controller
@AllArgsConstructor
public class ClientController {
    private IClientService clientService;
    @GetMapping(path = "/")
    public String home(){
        return "redirect:/user/client";
    }

    @GetMapping(path = "/user/client")
    public String clients(Model model,
                          @RequestParam(name = "page",defaultValue
= "0") int page,
                          @RequestParam(name = "size",defaultValue
= "5")int size,
                          @RequestParam(name =
"keyword",defaultValue = "")String keyword){
        Page<Client>
clients=clientService.getClientsByName(keyword,PageRequest.of(page,
size));
        model.addAttribute("listClients",clients.getContent());
        model.addAttribute("pages",new
int[clients.getTotalPages()]);
        model.addAttribute("allPages", clients.getTotalElements());
        model.addAttribute("current",page);
        model.addAttribute("size",size);
        model.addAttribute("keyword",keyword);
        return "clients";
    }
    @PostMapping(path = "/admin/addClient")
    public String addClient(@RequestBody Client client){
            clientService.addClient(client);
            return "redirect:/user/client";
    }
```

```
    @GetMapping(path = "/admin/deleteClient")
    public String deleteClient(long id){
        clientService.deleteById(id);
        return "redirect:/user/client";
    }
}
```

- **Gestion des Abonnement**

```
@Controller
@AllArgsConstructor
public class AbonnementController {
    private IAbonnementService abonnementService;
    private IClientService clientService;
    @GetMapping(path = "/user/abonnement")
    public String clients(Model model,
                          @RequestParam(name = "page",defaultValue
= "0") int page,
                          @RequestParam(name = "size",defaultValue
= "5")int size,
                          @RequestParam(name =
"keyword",defaultValue = "")String keyword){
        Page<Abonnement>
abonnements=abonnementService.getAbonnementsByMontant(keyword,PageR
equest.of(page,size));
        Page<Client>
clients=clientService.getClientsByName("",PageRequest.of(0,100));


model.addAttribute("listAbonnements",abonnements.getContent());
        model.addAttribute("listClients2",clients.getContent());
        model.addAttribute("pages",new
int[abonnements.getTotalPages()]);
        model.addAttribute("allPages",
abonnements.getTotalElements());
        model.addAttribute("current",page);
        model.addAttribute("size",size);
        model.addAttribute("keyword",keyword);
        return "abonnements";
    }
    @PostMapping(path = "/admin/addAbonnement")
    public String addAbonnement(@RequestBody Abonnement
abonnement){
            abonnementService.addAbonnement(abonnement);
            return "redirect:/user/abonnement";
    }
    @GetMapping(path = "/admin/delete")
    public String delete(long id){
        abonnementService.deleteById(id);
        return "redirect:/user/abonnement";
    }

}
```

3- Web service RESTful

```
4- @RestController
   @RequestMapping("/api")
   @AllArgsConstructor
   public class ClientAbonnementRestfullController {
```

```
    private IClientService clientService;
    private IAbonnementService abonnementService;

    @GetMapping("/client")
    public List<Client> clients() {
        return clientService.findAll();
    }

    @GetMapping("/client/{id}")
    public Client client(@PathVariable Long id) {
        return clientService.findById(id);
    }

    @PostMapping("/client")
    public Client save(@RequestBody Client client) {
        return clientService.addClient(client);
    }

    @PutMapping("/client/{id}")
    public Client update(@PathVariable Long id, @RequestBody Client
client) {
        return clientService.updateClient(id, client);
    }

    @DeleteMapping("/client/{id}")
    public void deleteClient(@PathVariable Long id) {
        clientService.deleteById(id);
    }
    @GetMapping("/abonnement")
    public List<Abonnement> abonnements() {
        return abonnementService.findAll();
    }

    @GetMapping("/abonnement/{id}")
    public Abonnement abonnement(@PathVariable Long id) {
        return abonnementService.findById(id);
    }

    @PostMapping("/abonnement")
    public Abonnement save(@RequestBody Abonnement abonnement) {
        return abonnementService.addAbonnement(abonnement);
    }

    @PutMapping("/abonnement/{id}")
    public Abonnement update(@PathVariable Long id, @RequestBody
Abonnement abonnement) {
        return abonnementService.updateAbonnement(id, abonnement);
    }

    @DeleteMapping("/abonnement/{id}")
    public void deleteAbonnement(@PathVariable Long id) {
        abonnementService.deleteById(id);
    }
}
```

5- Sécurité

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig  {
    @Bean
    public InMemoryUserDetailsManager inMemoryUserDetailsManager(){
        return  new InMemoryUserDetailsManager(
                // you can use "{noop}1111" to avoid crypting problem

User.withUsername("abdo").password("{noop}1111").roles("ADMIN","USER").bu
ild(),

User.withUsername("youssef").password("{noop}2222").roles("USER").build()
,

User.withUsername("oumaima").password("{noop}3333").roles("USER").build()
        );
    }
    /*@Bean
    PasswordEncoder passwordEncoder(){
        return new BCryptPasswordEncoder();
    }*/
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
httpSecurity) throws Exception {
        // all requests required authentification
        httpSecurity.formLogin().loginPage("/login").permitAll();

httpSecurity.authorizeHttpRequests().requestMatchers("/webjars/**").permi
tAll();

httpSecurity.authorizeHttpRequests().requestMatchers("/user/**").hasRole(
"USER");

httpSecurity.authorizeHttpRequests().requestMatchers("/admin/**").hasRole
("ADMIN");
        httpSecurity.exceptionHandling().accessDeniedPage("/403");

httpSecurity.authorizeHttpRequests().anyRequest().authenticated();

        return httpSecurity.build();
    }

}
```

```java
@Controller
public class SecurityController {

    @GetMapping("/403")
    public String notAuthorized(){
        return "403";
    }
    @GetMapping("/login")
    public String login(){
        return "login";
    }
}
```