**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**



LUCRARE DE LICENȚĂ

**Hotel Manager**

**propusă de**

***Adrian Ștefan***

**Sesiunea:** *iulie, 2019*

**Coordonator științific**

Colaborator Florin Olariu

**UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI**

**FACULTATEA DE INFORMATICĂ**

Hotel Manager

*Adrian Ștefan*

**Sesiunea:** *iulie, 2019*

**Coordonator științific**

*Colborator Florin Olariu*

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Data \_\_\_\_\_\_\_\_\_\_\_\_ Semnătura \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**DECLARAȚIE privind originalitatea conținutului lucrării de licență**

Subsemntatul(a) ………………………………………………………………………………………

domiciliul în …………………………………………………………………………………………………..

născut(ă) la data de ………………..…., identificat prin CNP ………….……………..………………..., absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de ………………………. specializarea …………………………………………………………, promoția …………………………., declar pe propria răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 si 5 referitoare la plagiat, că lucrarea de licență cu titlul: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_elaborată sub îndrumarea dl. / d-na \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_, pe care urmează să o susțină în fața comisiei este originală, îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice in vederea facilitării fasificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Dată azi, ………………………… Semnătură student …………………………

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „*Hotel Manager*”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de testetc.) 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, *24.06.2019*

Absolvent *Adrian Ștefan*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Cuprins**

Introducere5

Contribuții6

Capitolul 1. Descrierea problemei7

1.1 Tehnologii folosite7

1.2 Arhitectură proiect11

1.3 Bune practici folosite14

Capitolul 2: Abordări anterioare16

Capitolul 3: Descrierea soluției17

3.1 Înregistrare și logare17

3.2 Securitate18

3.3 Rezervare cameră20

3.4 Comandă room-service23

3.5 Activități admin25

3.6 Algoritm preț variabil26

3.7 Serviciu emailuri32

3.8 Plată PayPal34

Concluzii37

Bibliografie38

**Introducere**

Tema licentei este construirea unei aplicații web care are ca și scop managemntul unui hotel. Toți utilizatori aplicației vor avea posibilitatea să vadă informații generale despre hotel și serviciile oferite de acesta. Principalele utilități ale aplicației pentru client sunt cea de rezervare a camerelor și de plasare a comenzilor de tip room-service. Accesul spre serviciile de rezervare de camere si comenzi room-service se face prin autentificarea cu un cont personal.

Aplicația oferă posibilitatea înregistrări cu trei tipuri de conturi, admin, angajat și utilizator. Ca și admin poți crea diferite categorii de camere. Odată creată o categorie poti adăuga una sau mai multe camere cu categoria respectivă, acestea devenind disponibile spre rezervare. De asemenea, admini au și posibilitatea de a adauga diverse tipuri de produse pentru a putea fi comandate prin serviciul de room-service. Rezervarea camerelor poate fi facută de toți utilizatorii logați și plata se face prin PayPal pentru finalizarea rezervării. Comenzile room-service pot fi făcute de utilizatorii care sunt logați și au o rezervare de camera în momentul efectuării comenzii. Conturile de tip angajat pot vedea toate rezervările și comenzile de room-service. In plus, aceștia pot face rezervări și comenzi de room-service pentru clienți care nu au cont personal, plata pentru aceste rezervări poate fi de orice tip, nu doar PayPal.

O altă utilitate importantă aplicației este cea de generare a unui cod QR la momentul înregistrării. Cu acest cod de bare clienții vor putea intra în camerele unde au rezervări pe parcursul rezervării. Fiecare camera a hotelului va avea un dipozitiv capabil să scaneze codul QR, iar la scanarea codului corect se va descuia camera. Codul QR necesar descuierii camerei este actualizat în funcție de rezervarea activă pentru acea camera.

Motivul pentru care am ales această temă de licență este că, deși există multe hoteluri cu aplicații care să ofere posibilitatea rezervării unei camere, nu am găsit nici o aplicație care să faciliteze toată experiența ta la acel hotel prin intermediul aplicației. La aceste hoteluri chiar dacă îți faci rezervare la cameră online odată ajuns la hotel tot trebuie să mergi la recepție pentru a primi cheia camerei ceea ce în anumite cazuri poate duce la timp pierdut prin așteptarea la cozi lungi la recepție. Alt element care nu l-am văzut la nici o altă aplicatie este posibilitatea plasării unei comenzi de room-service, și consider ca este mai mult pe placul lumii să plaseze o comandă prin internet și să poată vedea statusul comenzii decât prin intermediul telefonului sau în alte moduri.

De asemenea, aplicația foloseste și algoritmi pentru a face prețul rezervării să varieze în funcție de diverse criterii, printre care și oferirea de reduceri în funcție de rezervările anterioare.

Așadar, consider că cel mai mare plus adus de aplicația mea este oferirea posibilității cazării și cererii serviciilor la un hotel doar prin intermediul aplicației, asta făcând atât cazarea cât și perioada de cazare la hotel mult mai simplă și mai plăcută pentru client

**Contribuții**

Contribuțiile aduse de această aplicație sunt:

* Posibilității de a te caza la hotel doar prin intermediul aplicației, după rezervare primind un QR cod cu care poți intra în cameră, nefiind nevoie să mergi la recepția hotelului pentru a îți face check-in-ul și pentru a ridica cheia;
* Posbilitatea de a da comenzi de room-service prin intermediul aplicației, statusul comenzilor fiind actualizat constant de către angajați hotelului, clienții stiind exact în ce status este comanda lor;
* Camerele și alimentele disponbilie sunt configurabile prin intremediul aplicației de către un utilizator de tip admin;
* Folosirea unui algoritm pentru a genera prețul unei rezervării variabil după mai multe criterii.

Ca și contribuții personale, pentru dezoltarea acestui proiect am învățat să folosesc două frameworkuri, unul pentru dezoltarea interfeței grafice (Angular) și unul pentru dezoltarea back-end-ului (Spring). Pe langă asta, am învățat și cum să folosesc sistemul extern pentru realizarea unei plăți prin PayPal și să trimit emailuri cu sistemul Gmail.

**Capitolul 1. Descrierea problemei**

**1.1 Tehnologii folosite**

Proiectul este împărțit în două aplicații, o aplicație este un API care se ocupă de prelucrarea informațiilor, cât și stocarea si preluarea informațiilor in baza de date, această parte este numită partea de back-end. A doua aplicație a proiectului este interfața grafică a proiectului, aceasta făcând apeluri spre partea de back-end și afisându-le în interfața grafică, această parte este numită front-end.

**Back-end**

Tehnologiile folosite pentru back-end sunt limbajul de programare Java 8, frameworkul Hibernate pentru lucrul cu baza de date și frameworkul Spring. Baza de date folosită este PostgreSQL. Pentru aducerea de dependințe externe în proiect este folosit sistemul Maven. Spre exemplu, pentru folosirea frameworkurilor este folosit Maven. Pentru a folosi Maven se creează în proiect un fișier numit “pom.xml”. În acest fișier se pot face mai multe configurări, dar stric pentru aducerea de dependințe externe este folosit tagul <dependencies> </dependencies>, în interiorul căruia se adaugă diverse dependințe preluate de pe internet (vezi figura 1).

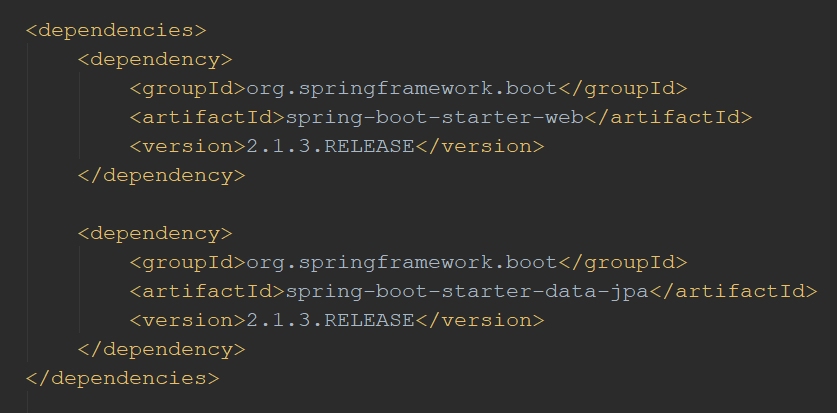


Figura 1: Importare dependința folosind maven

Hibernate este un framework care se ocupă cu maparea claselor folosite în cod la tabele în baza de date, astfel lucrând cu obiecte și clase direct pentru a prelua și adauga informații în baza de date. Pentru a folosi acest framework este folosit Java Persistence API, numit JPA, aceasta fiind specificția unei interfețe de programare a aplicațiilor care descrie managementul datelor relaționale în aplicații foloșind Hibernate pentru lucrul cu baza de date. JPA foloseste diverse adnotări și interfețe pentru lucrul cu baza de date. Unele din cele mai importante adnotări sunt @Entity folosită asupra unei clase pentru a mapa acea clasă la o tabelă în baza de date (vezi figura 2), @OneToOne, @OneToMany, @ManyToOne, @ManyToMany pentru a face legături între diferite tabele și adnotarea @Transactional pentru a specifica ca într-o anumită metodă sau clasă să se execute codul sub formă de tranzacție. De asemenea, este necesar ca această clasă să aibă metode de setare și aducere de informații (get și set).

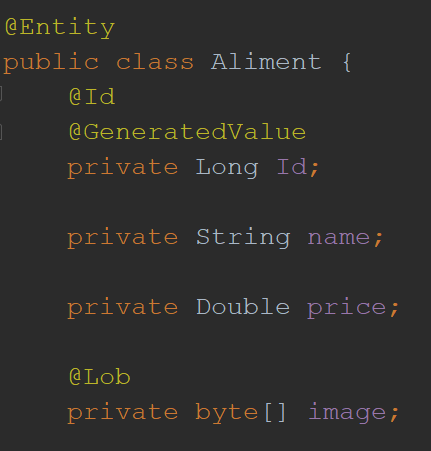


Figura 2: Mapare clasă la tabelă în baza de date folosind JPA

Spring este un framework cu multe utilități, cele mai importante utilități folosite în proiectul meu sunt Spring Data, Spring Boot, mecanism de inversiune de control, mecanism pentru cereri web de tipul verbelor HTTP cunoscute și Spring Security.

Spring Data oferă mai multe funcționalități peste JPA, adică multe metode deja implementate pentru preluarea și stocarea infromațiilor în baza de date. În proiectul meu am folosit JpaRepository (exemplu in figura 3), aceasta este o interfață cu diferite implementări standard pentru lucrul cu baza de date, spre exemplu findById(id) pentru preluarea obiectului cu id-ul cerut din baza de date, findAll() pentru preluarea tuturor obiectelor din tabela mapată la clasa cerută, save(obiect) pentru adăugarea unui obiect în baza de date. De asemenea, JpaRepository oferă posibilitatea de a crea întrebări specifice la baza de date prin crearea unei metode în JpaRepository deasupra la care este pusă adnotarea @Query(“”) în interiorul căreia este scrisă întrebarea specifică dorită spre baza de date.

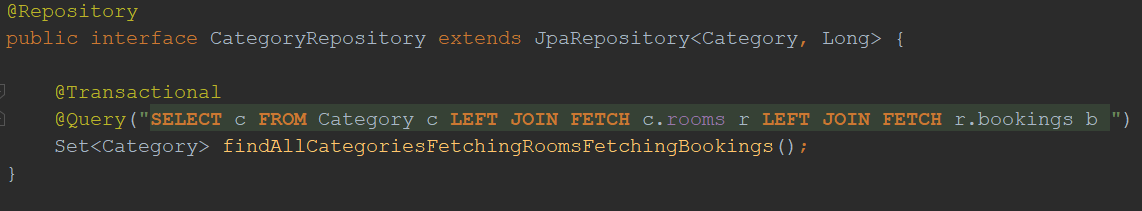


Figura 3: Exemplu interfață JpaRepository pentru interacționare cu baza de date

Spring Boot este un mecanism folosit pentru crearea configurațiilor aplicației și pentru rularea aplicației (vezi exemplu în figura 4).

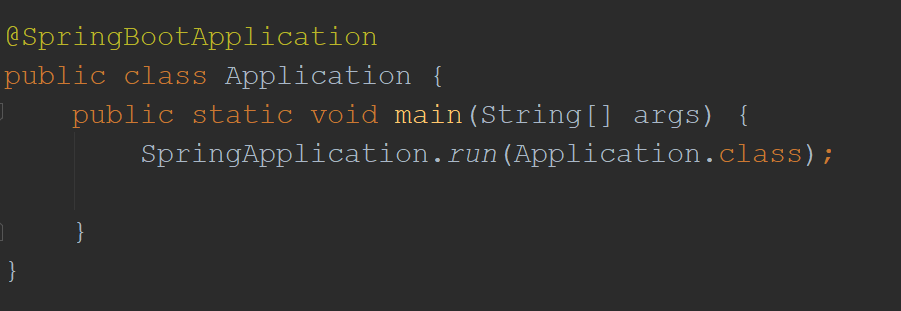


Figura 4: Clasa necesară pornire aplicație spring boot

Mecansimul de inversiune de control este realizat prin injectare de dependințe. Pentru a putea injecta un serviciu trebuie declarată o interfață și pusă o adnotare de tip @Service peste o clasă care implementează acea interfață dupa care această clasă trebuie declarată în constructorul clasei și deasupra constructorului trebuie pusă adnotarea @Autowired (vezi figura 5).

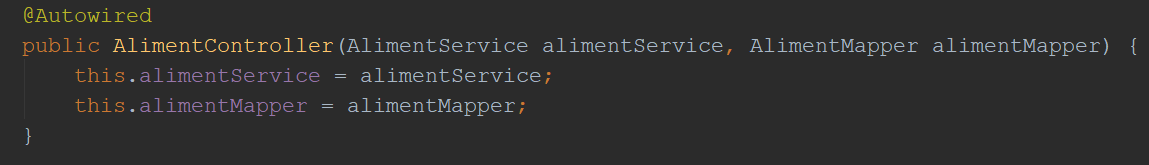


Figura 5: Exemplu injectare servicii

Mecanismul pentru cereri web este realizat prin diverse adnotări pentru a zice ca o metoda este o reprezentare pentru un verb HTTP. Pentru a defini o clasă ca o clasă cu verbe HTTP este pusă adnotarea @RestController deasupra ei și adnotarea @RequestMapping(“”) pentru a zice calea la care vom găsi verbele HTTP definite în clasă. Adnotările @PostMapping, @GetMapping, @PutMapping, @PatchMapping sunt folosite deasupra unei metode din clasă pentru a marca acea metodă ca unul din verbele HTTP. Pe lânga aceste adnotări mai sunt folosite și adnotările @PathVariable, @RequestBody, @ReqeustParam pentru folosirea de varabile în URL, informații transime in corpul cererilor web sau ca parametri.



Figura 6: Metoda care implementează getById folosind spring

Spring Security este folosit pentru autentificarea și securizarea aplicației de back-end prin generarea de tokeni la autentificare și folosirea acestor tokeni pentru a accesa diferite cereri web în funcție de gradul lor de securitate.

De asemenea, pe partea de back-end folosesc și Junit pentru testarea unitară a anumitor părți de cod care sunt mai complicate.

**Front-end**

Tehnologiile folosite pentru front-end sunt frameworkul Angular 7, limbajul de programare TypeScript, limbajul de scripting HTML5 si limbajul de modelare CSS3. De asemenea, sunt folosite și serviciile Bootstrap, Angular Material și PrimeFaces pentru folosirea de elemente de interfață grafică deja definite.

Angular este un framework care folosește ca bază module, componente și servicii. Modulele sunt folosite pentru organizarea codului, acestea fiind compuse din componente, servicii și/sau pipeuri. Componentele sunt create dintr-un fișier HTML, unul CSS si unul TypeScript din care se formează o pagină sau o parte a unei pagini web. Fisierele HTML și CSS oferă interfața grafică și fisierul TypeScript face apleuri folosind servicii la aplicația de back-end pentru preluare de date din baza de date și apoi prelucrarea acestora pentru a fi afișate în interfața grafică. De asemenea, modulele sunt încărcate prin tehnica de lazy loading, astfel oferind o performanța mai bună aplicației.

**1.2 Arhitectură proiect**

Proiectul are o arhitectura pe trei nivele, prezentare, aplicație și date.

Nivelul prezentare este interfața grafică a proiectului, deci acesta este reprezentat de aplicația de front-end. Acesta face cereri la aplicația de back-end și trimite informații browserelor în forma de HTML și CSS folosind frameworkul Angular.

Nivelul aplicație este aplicația de back-end. Aceasta folosește controllere pentru a prelua cererile web de la aplicația de front-end, apoi folosește servicii penru a prelucra informațiile și a adauga sau pentru a prelua anumite informații din baza de date. Sunt folosite două sisteme externe, unul pentru plata prin PayPal și unul pentru trimiterea de emailuri prin Gmail (vezi figura 7).

Nivelul de date este reprezentat de baza de date, care se ocupa de păstrarea și recuperarea informatiilor de către utilizator. Baza de date folosită in proiect este PostgreSQL.

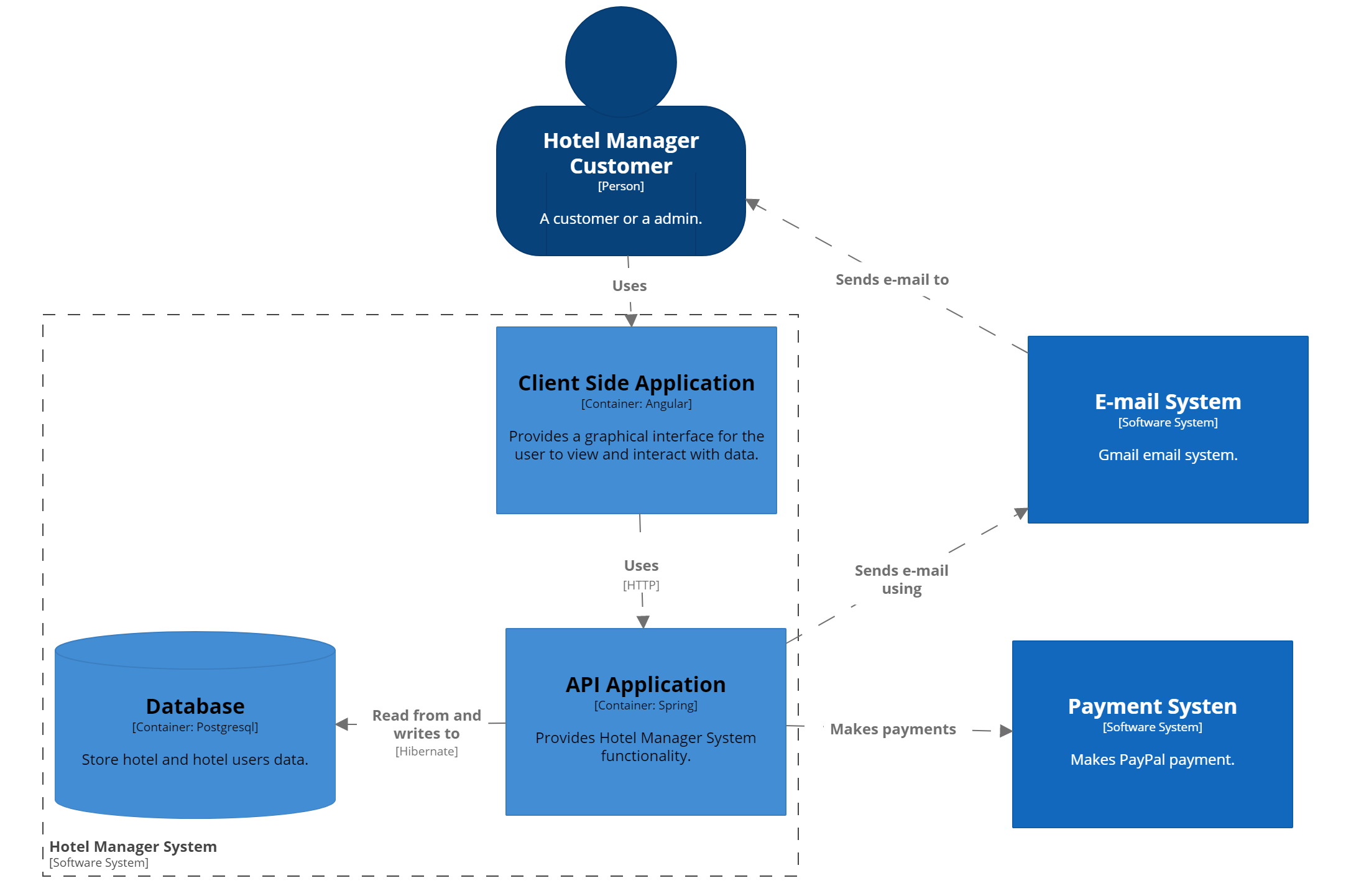


Figura 7: Diagramă de containere

**Back-end**

Aplicația de back-end este împarțită la rândul ei pe trei nivele, nivelul de preluare a cererilor web de la aplicația de front-end folosind controllere, mai departe folosind servicii pentru prelucararea datelor, serviciile reprezentând al doilea nivel. Ultimul nivel constă în lucrul cu bazele de informații, adică preluarea informațiilor și adăugarea acestora în baza de date, prin intermediul interfeței JpaRepository. Nivelul de lucrul cu bazele de date este folosit de nivelul de servicii. Atât folosirea serviciilor cât și a interfețelor de lucru cu baza de date se face prin injectare de dependințe.

Pentru lucrul cu baza de date se folosesc modele mapate la tabele din baza de date (la figura 8 se poate vedea diagrama de claselor de modele a aplicației).

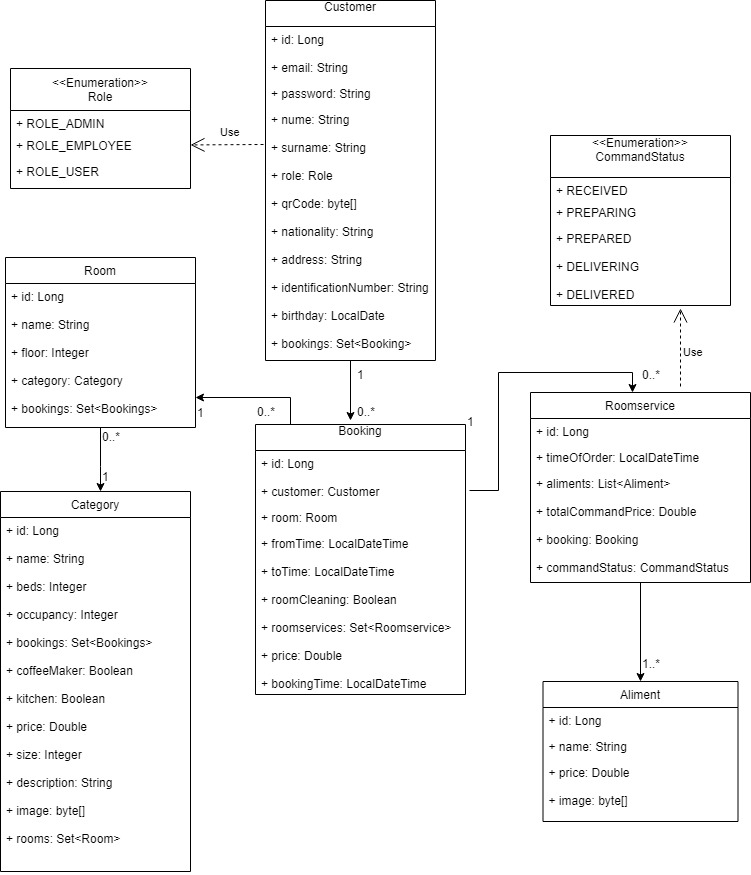


Figura 8: Diagrama claselor de modele

**Front-end**

Aplicația de front-end este împărțită în trei module, core, modules și shared. În modulul core se găsesc modelele pentru lucrul cu date primite de la back-end, serviciile care fac cererile web și gărzile necesare pentru securitate în funcție de gradul de acces al utilizatorului. Modulul modules conține la rândul lui diferite module organizate care conțin componente necesare pentru a crea o pagina web. În final, ultimul modul este shared care conține informații comune pentru celelalte module cum ar fi componentele cu header și footer, pipeuri serviciu pentru validări și aducere de diverse module externe.

**Securitate**

Pentru partea de securitate este folosit Spring Security pentru a genera un token la autentificare în funcție de rolul utilizatorului (admin pentru utilizatori de tip admini, employee pentru utilizatori care sunt angajati sau user pentru utilizatori normali care folosesc aplicația pentru a face o rezervare de cameră sau pentru a comanda room-service). Apoi sunt configurate cererile web pentru a oferi securitatea necesară în funcție de caz, securitatea acestora fiind verificată dupa informațiile din token.

**1.3 Bune practici folosite**

**Folosirea obiecte tip DTO**

Pentru preluarea de informații din cererile web și pentru returnare de informații sunt folosite alte clase asemănătoare cu modelel din baza de date, numite DTO (vezi figura 9). Motivul folosirii acestor clase este dat de faptul ca modelele trebuie să aibă aceeași formă ca și tabela din baza de date iar uneori putem să avem nevoie să primim sau sa oferim informații diferite fața de cele din model. Pentru a face conversia dintre aceste clase, adică de la model la DTO și invers, este folosit câte un serviciu numit mapper care conține metodele necesare pentru a mapa la tipul care avem nevoie. Procesul de mapare este făcut prin crearea unui obiect de noul tip și instanțierea acestuia cu valorile din obiectul dinspre care este mapat (vezi figura 10).

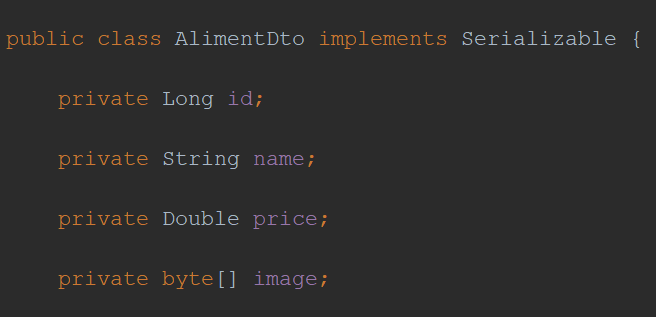


Figura 9: Exemplu clasă tip DTO

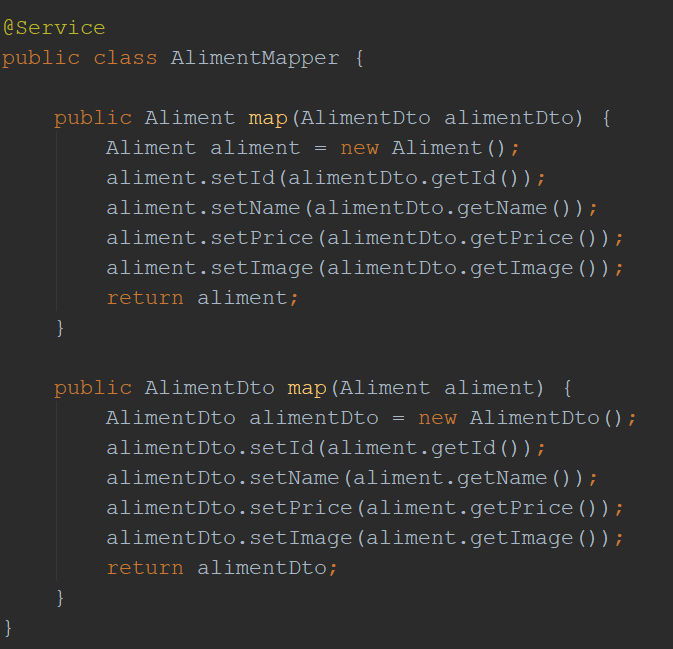


Figura 10: Exemplu serviciu de mapare de la dto la model

**Folosirea excepții proprii**

Aplicația folosește excepții proprii care extind clasa RuntimeException și le este dat un cod de status HTTP. Spre exemplu, dacă se caută o camera după un id în baza de date și nu este găsită acea cameră este aruncată excepția RoomNotFoundException cu codul de status 404.

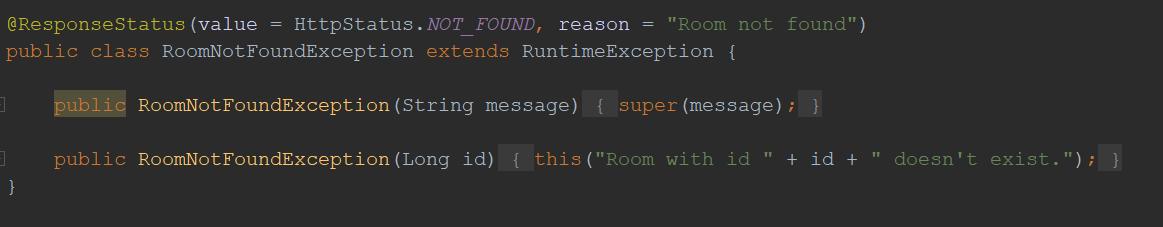


Figura 10: Excepție proprie mapată la un cod de status

**Responsivitate aplicație**

Aplicația este făcută responsivă să poată fi folosită pe orice tip de dispozitiv (spre exemplu calculator, tabletă, telefon).

**Clean code**

În scrierea codului sunt respectate tehnicile de clean code, adică:

* Numele funcțiilor și al variabilelor sunt descriptive pentru a fi ușor de înteles care este rolul acestora;
* Codul este scris cât mai modular posibil pentru a fi ușor de înteles și pentru eliminarea codului duplicat;
* În cod este păstrată consistența;
* Codul este organizat în pachete și servicii.

**GDPR**

Aplicația respectă Regulamentul General Privind Protecția Datelor prin codificarea datelor introduse în baza de date și decodificare acestora la preluarea datelor din baza de date. Pentru o securitate și mai buna pentru parolă este folosită o funcție hash înainte de a fi introdusă în baza de date.

**Capitolul 2. Abordări anterioare**

În practică există multe hoteluri care au aplicații făcute pentru a putea face rezervări la aceste hoteluri, dar mi se pare că cel mai mare minus la aceste aplicații este faptul ca deși oferă posibilitatea de a îți rezerva o cameră prin intermediul aplicației, odată ajuns la hotel pentru a te caza în cameră este necesar să mergi la recepție pentru a face check-in-ul și pentru a lua cheia ceea ce consider că este ceva foarte impractic și nu în ton cu evoluția tehnologiei în ziua de azi.

Un alt element care nu l-am găsit în nici o altă aplicație este posibilitatea de a face o comandă de room-service la care să poți vedea statusul comenzii în timp real.

Un plus care unele aplicații le au și nu am ajuns să implementez la aplicația mea este cel de a oferi posibiltatea de a face rezervări la anumite săli și la evenimente de tip spa și masaj prin intermediul aplicației.

Mai se găsesc niste exemple de aplicații ale unor hoteluri existente:

* <https://www.hotelinternationaliasi.ro/>
* <https://www.portolahotel.com/>
* <https://www.thelucernehotel.com/>
* <https://www.lafondasantafe.com/>

**Captiolul 3. Descrierea soluției**

**3.1 Înregistrare și logare**

Inițial o persoană care accesează această applicație poate vizualiza informații generale despre hotel și poate vedea informații despre toate tipurile de camere ale hotelului. Pentru a face o rezervare la o cameră clientul va trebui să își creeze un cont personal în aplicție, în caz că nu are deja. La crearea contului clientului i se vor cere diverse informații personale necesare pentru crearea actelor necesare de hotel la cazarea clientului. Peste parola utilizatorului se va folosi o funcție hash înainte de a se introduce în baza de date, iar peste celelalte informații sensibile despre utilizator se va utiliza un algoritm de codificare înainte de a fi introduse în baza de date. De asemenea, utilizatorul va trebui să confirme că este de acord cu termenii și condițiile hotelului. La finalul înregistrării utilizatorul va primi un email de confirmare la adresa de email introdusă.

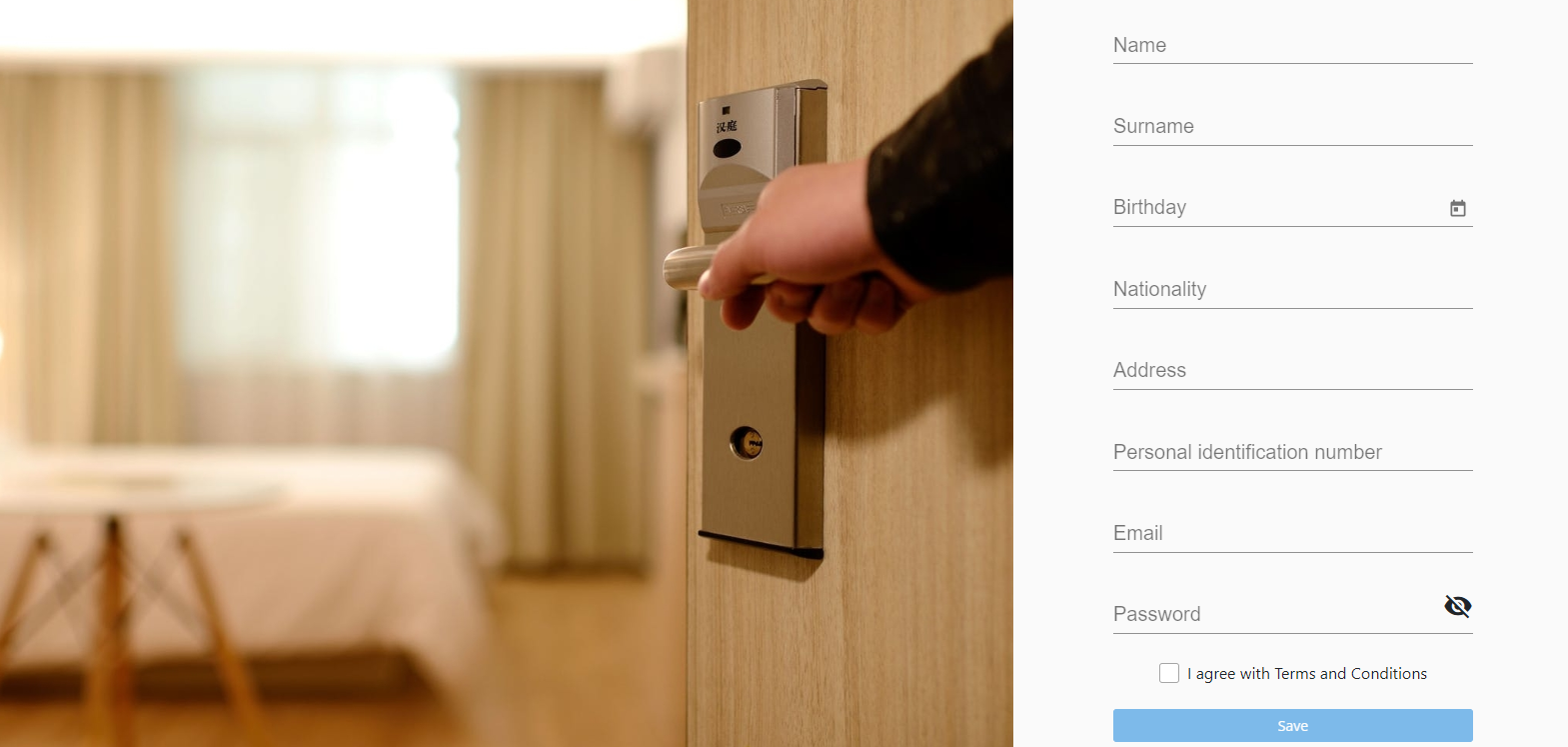


Figura 11: Pagină de înregistrare

Dupa finalizarea înregistrării utilizatorul va fi redirecționat catre pagina de autentificare, unde acesta trebuie să își introduca emailul și parola.

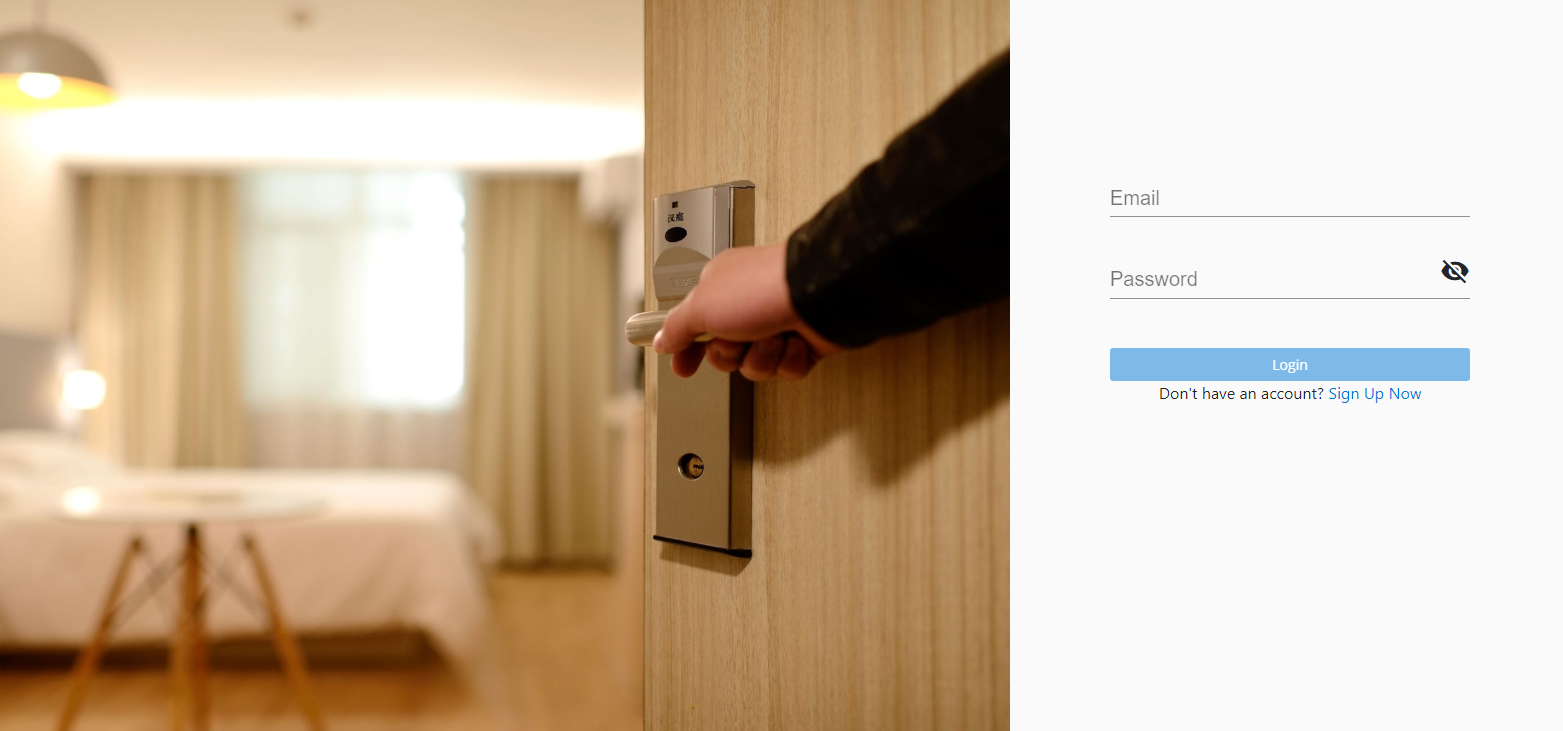


Figura 12: Pagină autentificare

**3.2 Securitate**

Pentru securitate se folosește Spring Security, prin introducerea dependinței spring-boot-starter-security în fișierul “pom.xml”. Apoi pentru activarea securității asupra aplicației a fost creată o clasă care să extinda WebSecurityConfigurerAdapter (vezi figura 13).

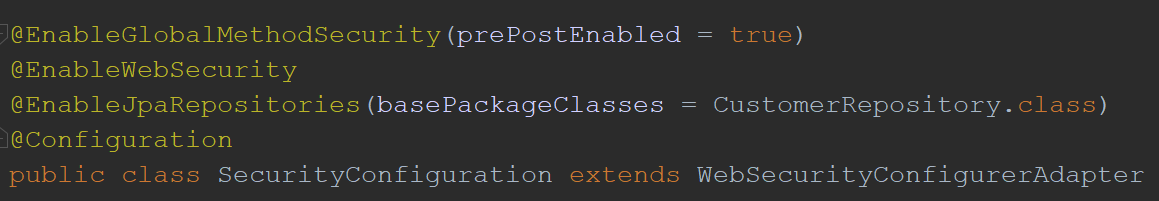


Figura 13: Clasă configurări pentru securitate

În această clasă este creat un bean AuthenticationManager care acum poate fi injectat în alte servicii. Mai departe acest bean va fi injectat în serviciul care se ocupă de autentificarea utilizatorilor și pe baza acestuia determină daca utilizatorul primește sau nu acces la aplicație.

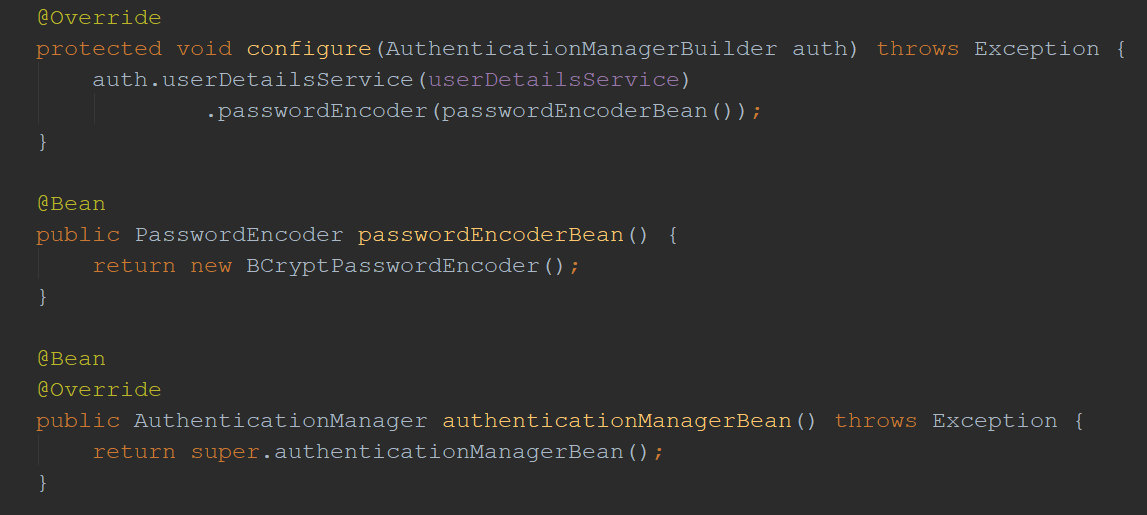


Figura 14: Configurare serviciu de autentificare

Beanul password encoder este cel care folosit pentru a aplica funcția hash peste parolă înainte de a fi introdusă în baza de date. De asemenea, dupa cum se vede tot acest bean este folosit și la configurarea serviciului de autentificare.

În cazul în utlizatorul este autentificat cu success se va genera un token care va fi trimis aplicației de front-end și îl va pastra în memoria acestuia locală (numită LocalStorage). Motivul păstrării tokenului în memorie este dat de faptul că odată ce este activat Spring Security asupra aplicției toate cererile web vor fi securizate și vor avea nevoie de un token valid pentru a fi accesate. Totuși, sunt nevoie și de cereri web care să se poată face și în cazul în care utilizatorul nu este logat, cum ar fi chiar cererea care este făcuta pentru logare, așdar tot în clasa de configurare trebuie să existe o metodă configure care în care se pot zice ce adresele cererilor web care nu necesită nici un fel de autentificare.



Figura 15: Configurări adrese care nu necesită nici un fel de autorizare pentru a le acesa

În plus, trebuie securizat accesul la anumite requesturi web în funcție de rolul utilizatorului. Spre exemplu, doar un utilizator cu rolul admin ar trebui să poată crea o cameră. Asta se face prin punerea adnotării @PreAuthorize(“hasRole(‘role\_name’)”) deasupra metodei care reprezinta cererea web pentru care se dorește securizarea în funcție de rol.

De asemenea, metode de securitate sunt folosite și pe partea de front-end a aplicației. Asta se va face prin folosirea unei clase de tip gardă, care la accesarea unei component va verifica dacă rolul utilizatorului este cel dorit.

**3.3 Rezervare cameră**

Pentru a rezerva o cameră la hotel un utilizator întâi trebuie să acceseze pagina cu toate categoriile de camere (vezi figura 16) disponibile spre rezervări, această pagină se poate accesa de pe pagina de acasă a aplicației unde se găsește o descriere a hotelului cât și a utilitătilor oferite de acesta sau din bara de meniu a aplicației la Accomodations.

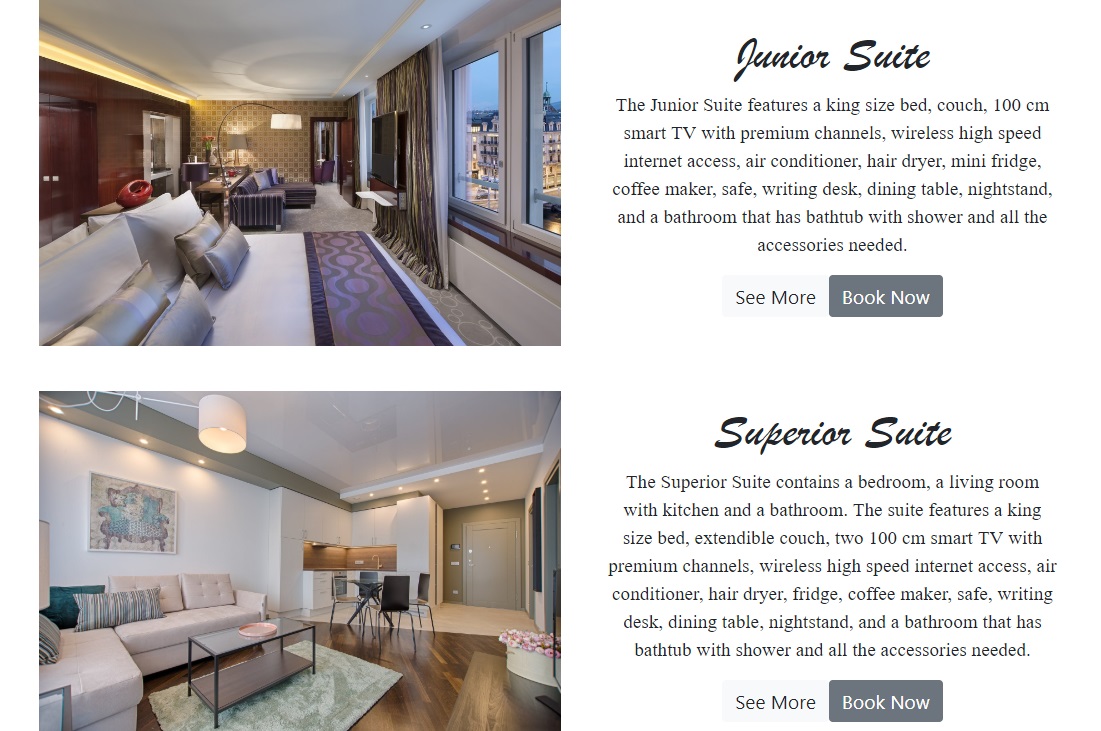


Figura 16: Pagină listare categorii de camere

De aici utilizatorul poate alege să rezerve o categorie de cameră apăsând butonul Book Now sau sa vadă mai multe informații despre categoria de cameră dorită.

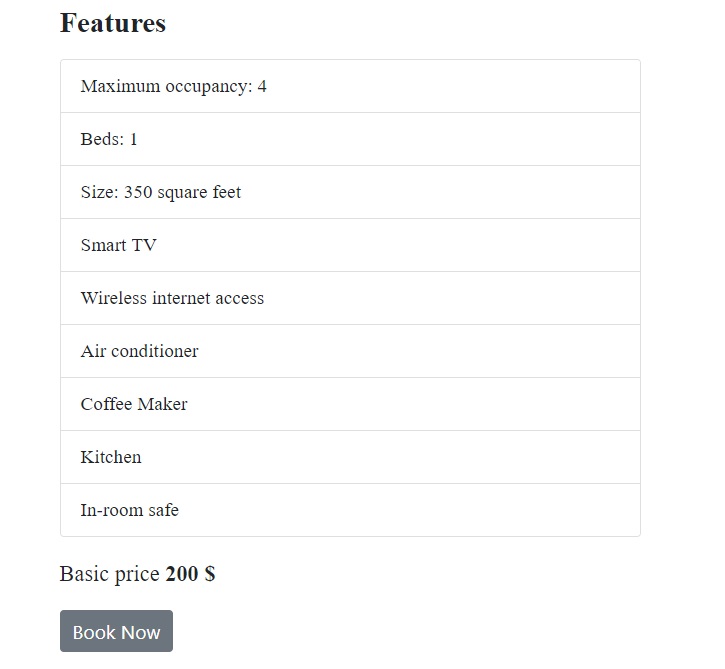


Figura 17: Informații categorie cameră

Dacă utilizatorul alege să rezerve o cameră acesta va fi dus la pagina din figura 18 unde va avea de ales data rezervării, adică ziua sosirii și ziua plecării.

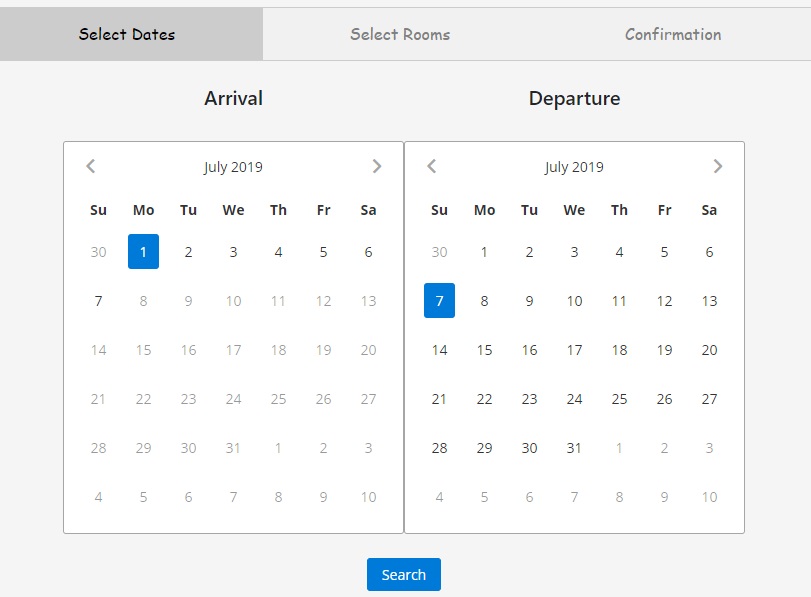


Figura 18: Pagină selectare periaodă rezervare cameră

Odata ce apasă butonul Search acesta va vedea categoriile de camere disponibile (vezi figura 19) pentru data rezervării aleasă și prețul complet pentru rezervarea dorită. Prețul este calculat variabil în funcție de un algoritm.

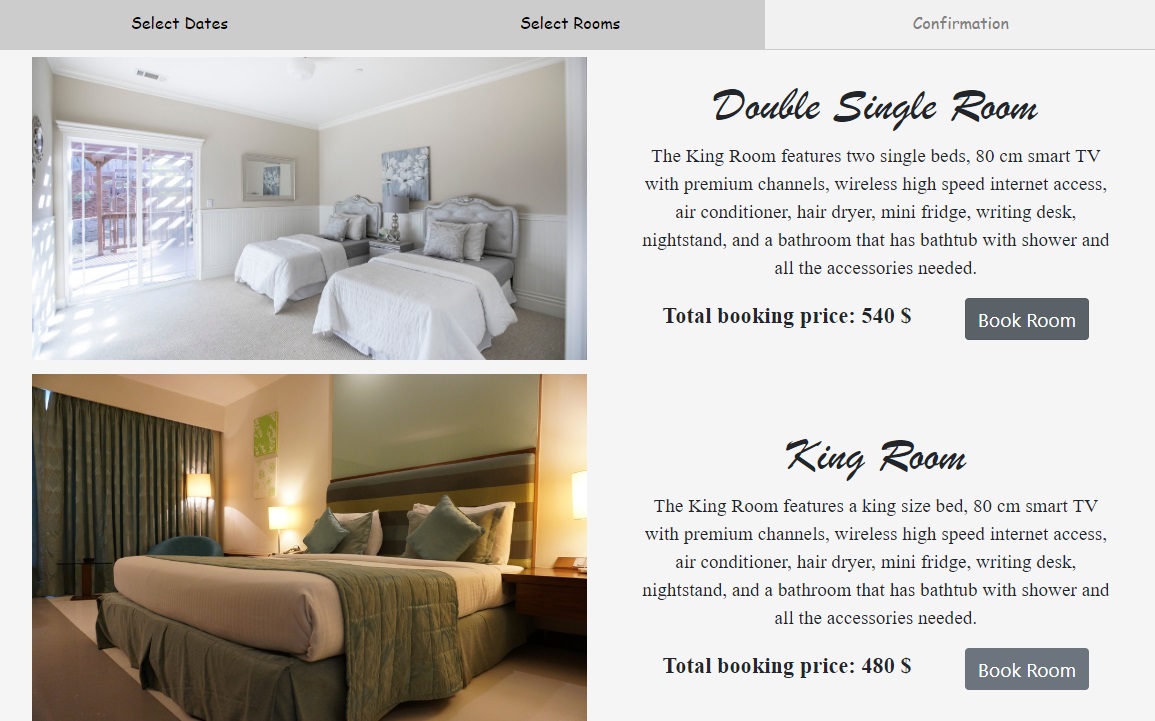


Figura 19: Pagină listare categorii disponibile în perioada de rezervare alesă și prețul rezervării

După ce vede categoriile de camere disponibile și pretul acestora utilizatorul va putea alege o cameră pentru a o rezerva în funcție de prețul și confortul dorit. După alegerea camerei dorite să fie rezervate acesta va fi dus la o pagină unde i se afisează informațiile pentru rezervarea dorită.

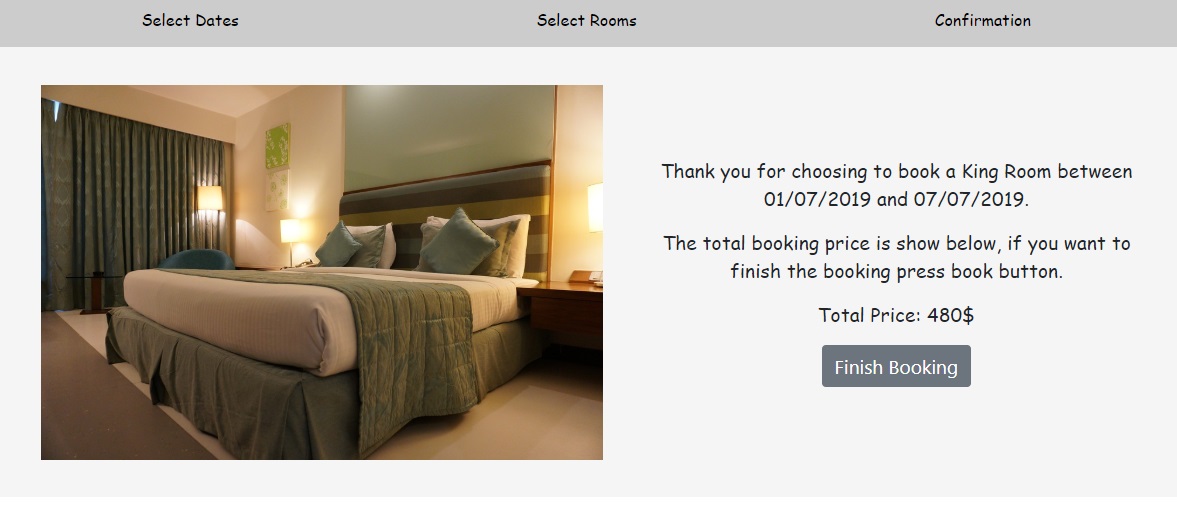


Figura 20: Pagină finalizare rezervare

În final, dacă îi convin datele rezervării utilizatorul va apăsa butonul Finish Booking și va fi redirecționat către o pagină pentru realizarea plătii printr-un sistem PayPal în cazul în care utilizatorul este un client. Pentru cazul în care un client vrea să facă o rezervare când ajunge la hotel și nu are un cont personal în aplicație sau nu vrea să plătească prin PayPal, clientul poate lua legătura cu un angajat iar acesta poate face rezervarea direct fără efectuarea plății prin PayPal și poate să încaseze banii lichizi direct de la client sau prin alt mod dorit de client.

La finalizarea comenzii se va trimite un email pentru a înștiința angajații hotelului că a fost facută o rezervare. În acest email se vor afla și informațiile necesare pentru crearea actelor închirierii camerei, acestea fiind date de utlizator la înregistrarea în aplicație.

De asemenea, fiecare utilizator îşi poate vedea o listă cu toate rezervările sortate descrescător în funcţie de data cazării. Utilizatorii de tip angajat vor putea vedea o listă cu toate rezervările făcute.

Apoi, în ziua cazării, tot ce va trebui să facă clientul pentru a se caza în camera sau camerele în care are rezervare este să descarce codul de bare de pe pagina de profil și să scaneze acest cod la intrarea în cameră pentru a descuia camera. Codul este valabil pe toată durata rezervării, adică din prima zi a rezervării de la 2PM până în ultima zi a rezervării la 12PM.

**3.4 Comandă room-service**

Pentru a face o comandă room-service un utilizator trebuie să acceseze pagina cu lista tuturor alimentelor posibile pentru comandă (vezi figura 21), această pagină se poate accesa de pe pagina de acasă a aplicației unde se găsește o descriere a hotelului cât și a utilitătilor oferite de acesta sau din bara de meniu a aplicației la Room-service.

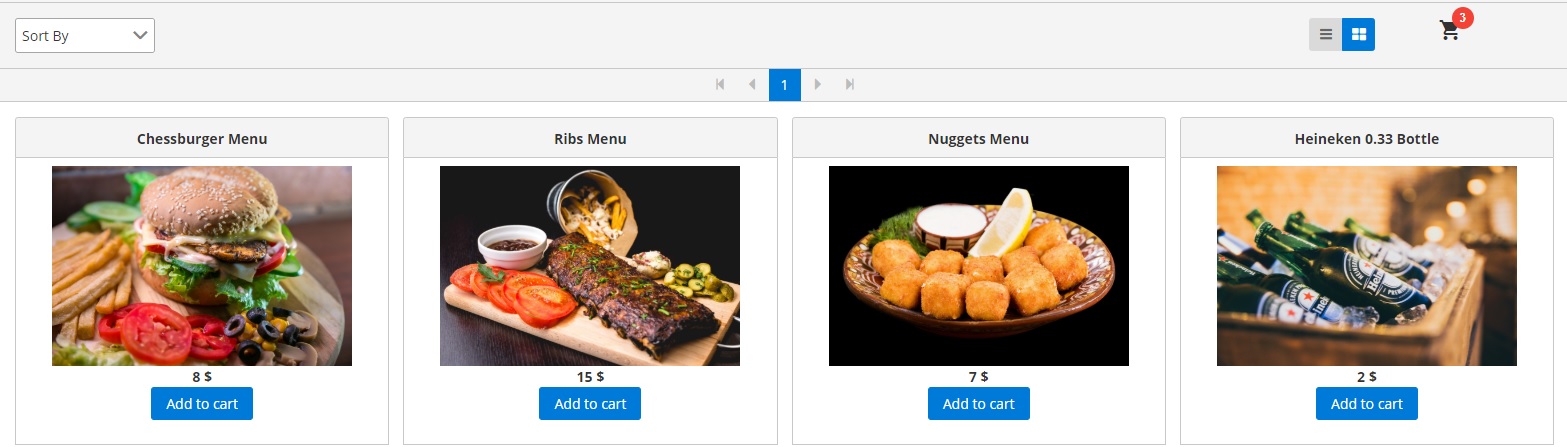


Figura 21: Pagină listare alimente posibile pentru comandă room-service

După ce ajunge aici utilizatorul poate să sorteze alimentele în funcție de preț pentru a găsi alimentele dorite de el. Apasând butonul Add to cart adaugă un aliment în coș. Pentru a finaliza comanda utilizatorul apasă pe coșul din dreapta sus, urmând să îi apară o listă cu alimentele alese și cu prețul final.

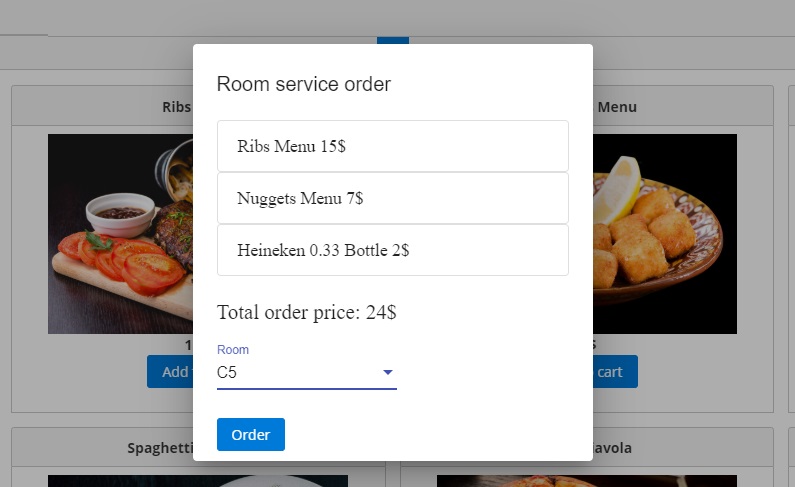


Figura 22: Modal coș care reprezintă o comandă room-service

Apoi dacă îi convine prețul și alimentele alese acesta trebuie să aleagă o cameră din camerele rezervate de el a caror rezervare se desfășoară în momentul actual dupa cum se vede în figura 22 (o rezervare activă fiind o condiție necesară pentru plasarea unei comenzi de room-service). De asemenea, dacă un client al hotelului dorește sa facă o comandă de room-service dar nu are cont personal va putea face comanda telefonic, urmând ca acești pași să fie făcuți de un angajat al hotelului. Dupa plasarea comenzii utilizatorul va fi dus la o pagină unde ii vor apărea tot istoricul comenzilor cu detalii despre acestea, ultima fiind poziționată cel mai sus (vezi figura 23).

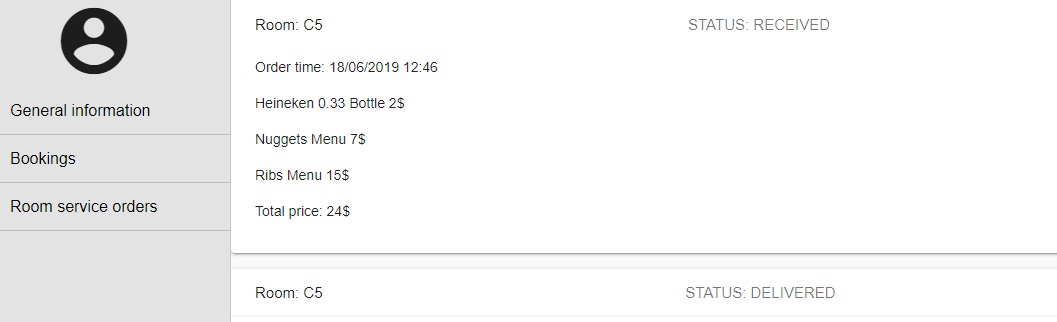


Figura 23: Pagină listare comenzi room-service ale utilizatorului

Un detaliu important este că în dreptul comenzii apare statusul acesteia pentru a informa utilizatorul în ce stadiu este comanda lui. La crearea comenzii acesta are statusul RECEIVED ceea ce înseamnă înregistrarea comenzii de către aplicație.

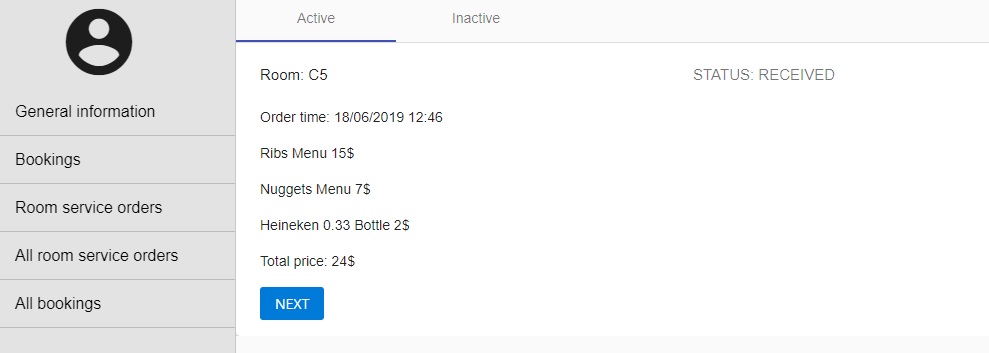


Figura 24: Pagină listare și management al tuturor comenzilor de catre angajați ai hotelului

Odată ce comanda a fost trimisă utilizatorii de tip angajat pot vedea comanda în lista cu toate comenzile active în momentul de față și ei vor trece comanda în următorul pas apăsând butonul Next din figura de mai sus, astfel actualizându-se statusul acesteia (vezi figura 24).

Următorii pași ai comenzii sunt:

* PREPARING: comanda a fost preluată de angajaţi şi au început preapararea acesteia;
* PREPARED: comanda a fost preparată de bucătari şi acum se aşteaptă preluarea comenzii de către personalul hotelului care aduce comanda în cameră;
* DELIVERING: comanda a fost preluată și este în process de livrare la camera clientului;
* DELIVERED: comanda a fost livrată în cameră clientului.

**3.5 Activități admin**

Utilizatorul de tip admin este cel care are privilegiile să creeze noi categorii de camere, noi camere pe baza categoriilor deja existente și noi alimente pentru comenzile de room-service. De asemenea, utilizatorul de tip admin este cel care are drepturi să creeze utilizatori de tip angajat sau admin. Initial este introdus în aplicație un utilizator de tip admin pentru a putea crea noi utilizatori de tip admin si angajat.

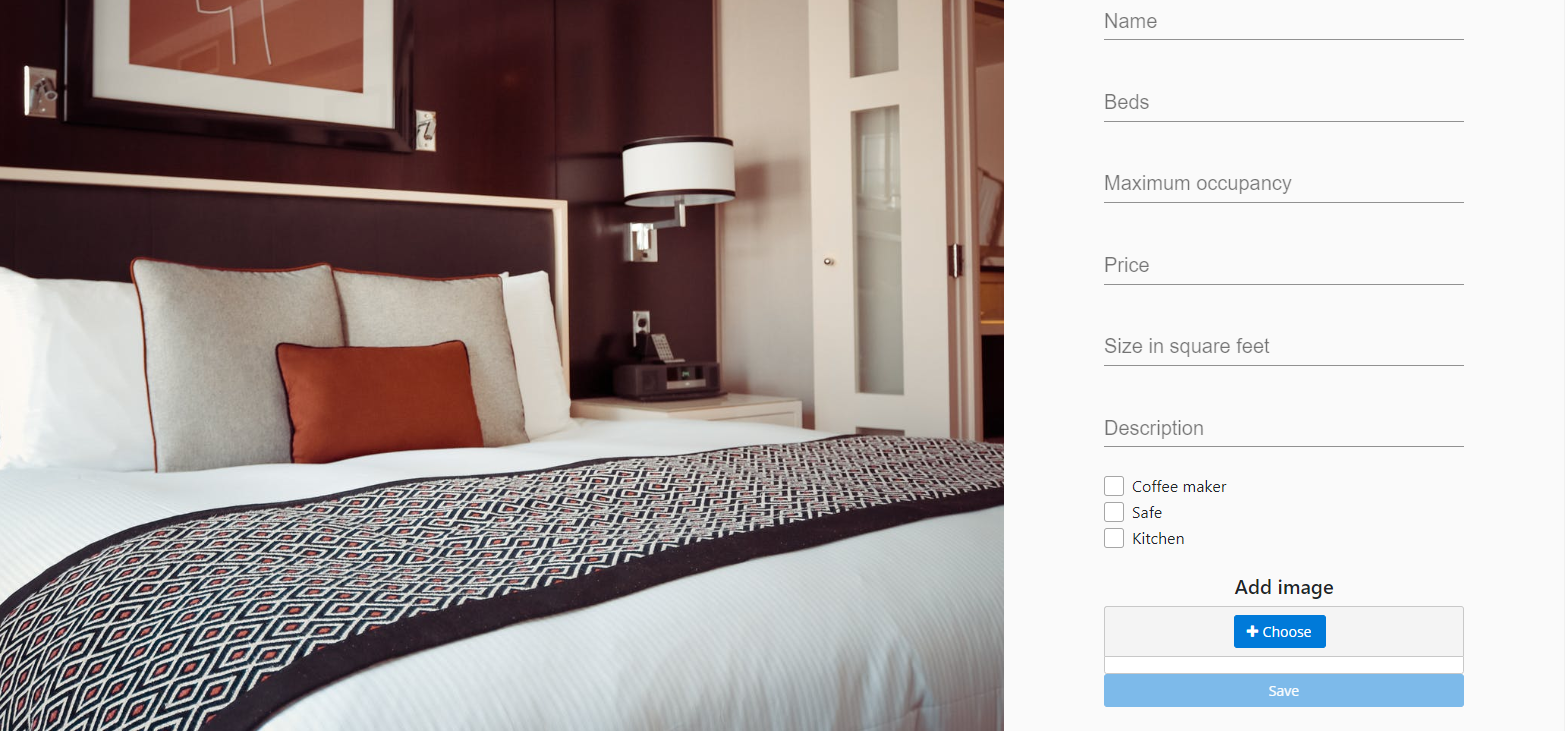


Figura 25: Pagină creare categorie de cameră, accesibilă utilizatorilor de tip admin

**3.5 Algoritm preț variabil**

Pentru afișarea prețului la rezervarea camerei se folosește un algoritm care să facă prețul variabil în funcție de mai multe criterii.

Criteriile după care este calculat prețul sunt:

* în funcție de procentajul de camere ocupate din categoria de camere prețul poate crește după câte camere sunt libere pe durata rezervării sau poate să scadă dacă procentajul de rezervare al categoriei de camere respective este foarte mic;
* prețul poate crește în funcție de anumite zile ale anului în care este știut că se fac multe rezervări;
* dacă rezervarea este pe un număr de zile mare prețul poate scădea în funcție de numărul de zile pe care se face rezervarea;
* în cazul în care în ultimele 24 de ore a fost făcut un număr mare de rezervări prețul camerelor va crește;
* prețul poate scădea dacă clientul a mai avut rezervări precendente la hotel în funcție de numărul de rezervări care le-a avut și în funcție de prețul total al rezervărilor.

Algortimul este implementat într-un serviciu separate definit în interfața PriceService (vezi figura 26), această interfață are o metodă care returnează un Map (o structură de tip dicționar) în care cheile sunt nume de categorii de cameră și valoarea este prețul pentru fiecare categorie de cameră, generat în funcție de categoriile pentru care se caută prețul, data sosirii, data plecării și emailul utilizatorului.

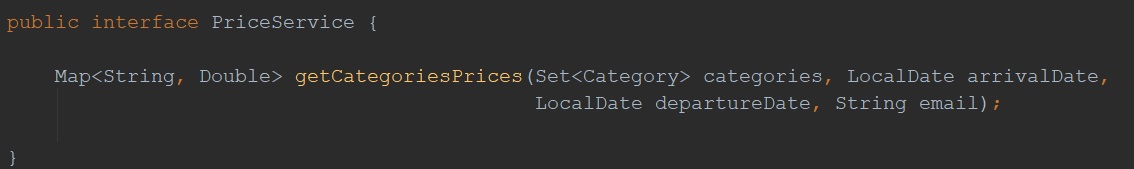


Figura 26: Interfața pentru serviciul care generează prețul rezervării

Această interfață este implementată de o clasă care suprascrie metoda din interfață și are ca membri o structură de tip map în care se vor pune categoriile cu prețul lor și serviciile pentru cameră și rervări care vor fi folosite pentru calcularea prețului categoriilor (vezi figura 27). Aceste două servicii sunt injectate în serviciul current prin procedeul de injectare de dependințe prin adnotarea @Autowired.

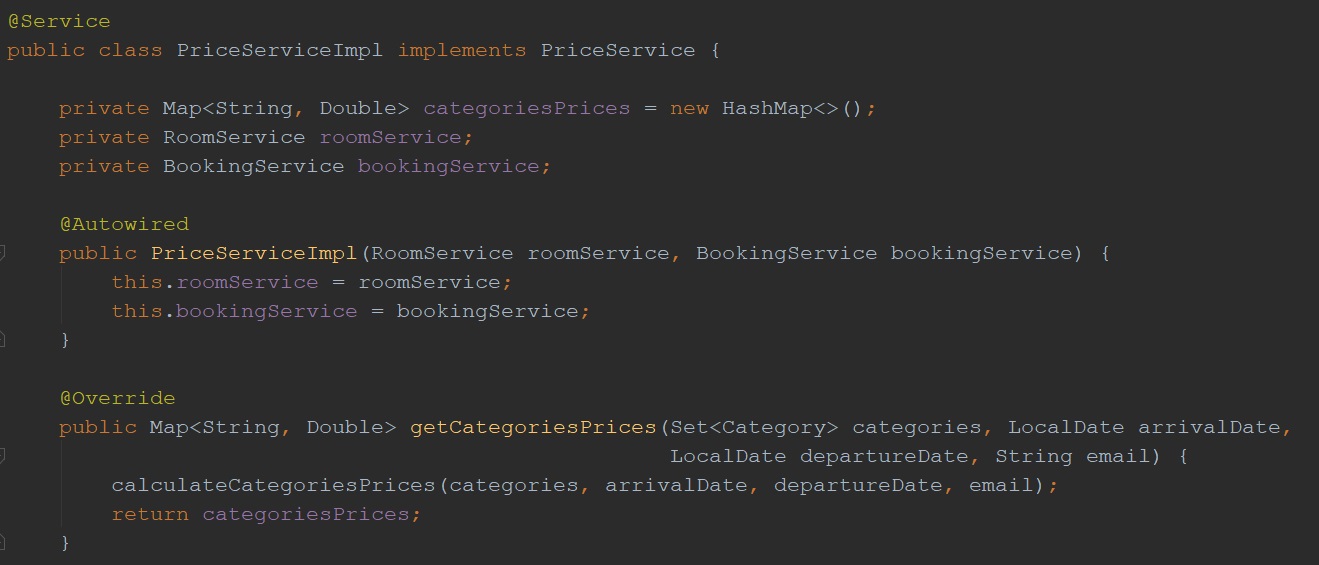


Figura 27: Inițializarea serviciului care se ocupă de generarea prețului rezervării

Metoda suprascrisă din interfață apelează altă metodă (calculateCategoriesPrices găsită în figura 28) a cărui scop este să pună în map la chei numele categoriilor de camere primate ca parametru de aceasta și pentru valoarea fiecărei chei apelează o metodă (calculateTotalBookingPrice găsită în figura 28) a cărui scop este să calculeze și să returneze prețul rezervării pentru o categorie de cameră care o primește la parametru.

Mai departe, metoda calculateTotalBookingPrice parcurge toate zilele rezervării și adaugă într-o variabilă prețul calculate pentru fiecare zi a rezervării de metoda getBookingPriceForDayByCategory (vezi figura 29). Apoi prețului curent i se vor aplică reduceri si majorări dupa urmatoarele metodele:

* getPriceRemainingPrecentageByNumberOfBookingDays (figura 33): calculeză un procetaj al prețului pe baza numărului de zile pentru care a fost făcută rezervare, dacă numărul de zile a rezervării este mare oferind o reducere a prețului;
* getPriceIncreasePercentageByNumberOfBookingInLastDay (figura 34): calculează un procentaj al prețului pe baza numărului de rezervări făcute în ultimele 24 de ore, dacă numărul de rezervări este mare majorând prețul;
* getPriceDiscountPercentageByCustomerPreviousBookings (figura 35): calculează un procentaj al prețului pe baza rezervărilor percedente a clientului care face cererea, oferind reduceri în funcție de numărul de rezervări precedente si de prețul total al rezervărilor (această reducere se aplică doar pentru clienții care fac rezervarea prin contul lor personal).

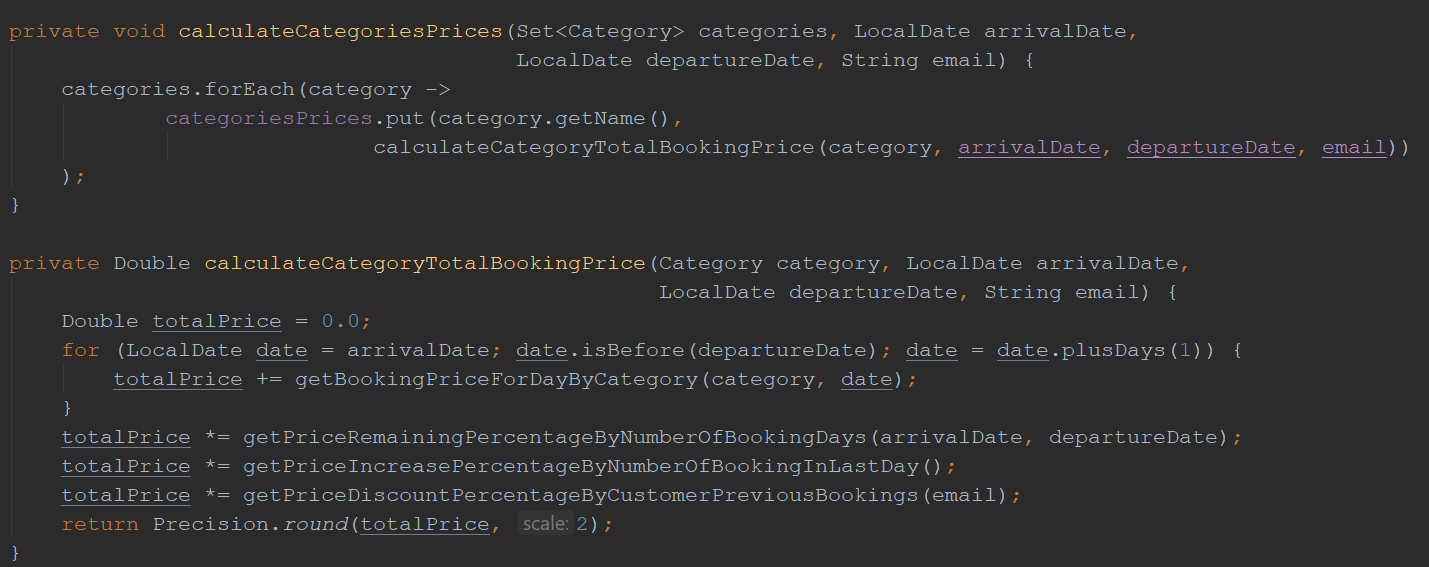


Figura 28: Metodă pentru inițializare map categorie-preț și metoda calculare preț total categorie

La apelarea metodei pentru obținere a prețului pentru o zi, acestă pune intr-o variabila prețul de bază al categoriei, după care adaugă prețului o valoare în funcție de numărul de camere ocupate apelând funcția getExtraPriceByRoomsOccupancy și înca o valoare dacă ziua respectivă se află pe lista de zile alese pentru majorare a prețului apelând funcțtia getExtraPriceBySpecialDates după cum se vede în tabelul de mai jos.

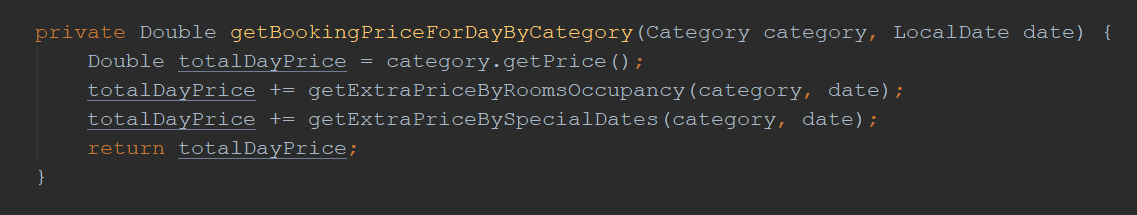


Figura 29: Metoda care calculează prețul categoriei pentru o anumită zi

Pentru calcularea adaosului de preț după procentajul de ocupare a categoriei, se va calcula procentajul de ocupare a categoriei de camere pentru ziua respective prin împărțirea numărului de camere ocupate la numărul total de camere din acea categorie. Pentru aflarea numărului de camere disponibile în acea zi se folosește metoda getAvailableRoomsBetweenDates din serviciul roomService. Apoi pentru calcularea adosului de preț se folosește o structură de tip enum în care într-o variabilă se vor găsi valorile la care dacă un procentaj de ocupare este mai mare sau egal decât valoare percent din enum atunci valoarea cu care se va multiplica pretul este valoare din campul multiplyValue. Astfel se parcurg toate valorile din enumul CategoryOcupancy iar în cazul în care procentajul de ocupare al camerelor actual este mai mare decât procentajul din enum vom returna valoarea pretului categoriei de camere înmulțită cu valoarea de multiplicare. De asemenea, avem și un caz special, adică daca mai este doar o zi până la rezervarea pentru ziua dorită si procentajul de ocupare este mic vom returna o valoare negativă să se adauge la preț. Vezi cele 2 tabele de mai jos pentru detalii.

Exemplu: dacă procentajul de ocupare al unei categorii camere este mai sau egal ca 0.8 atunci se va returna 0.2 înmulțit cu prețul de bază.

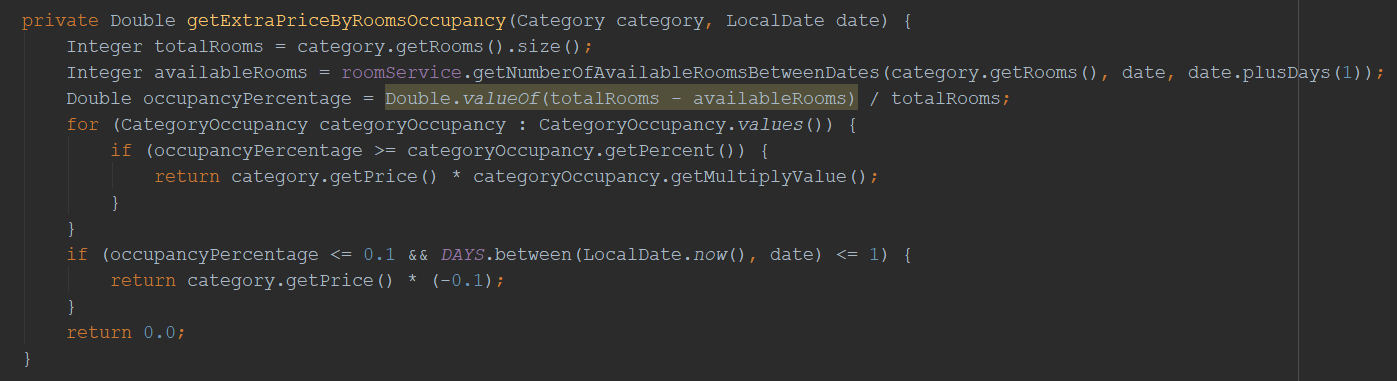


Figura 30: Metoda care calculează adaosul de preț al categoriei în funcție de nivelul de ocupare al categoriei într-o anumită zi

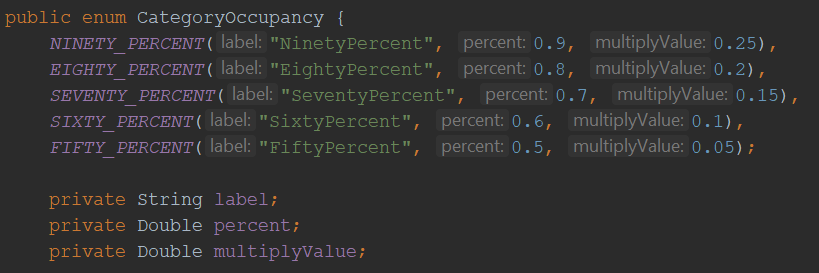


Figura 31: Enumul care este folosit pentru a calcula adaosul de preț pentru o zi în funcție de nivelul de ocupare

La calcularea adaosului de preț dupa zile special se va folosi în mod similar un enum în care se vor găsi ziua, luna și procentajul de multiplicare (vezi tabelul 32 pentru detalii suplimentare).

Exemplu: dacă pe în enum se gasește ziua 31, luna 12 si procentajul de multiplicare 0.3 și ziua pentru care se face rezervarea este 31/12/2019 atunci valoarea returnată de această funcție este valoarea prețului de bază înmulțită cu 0.3.

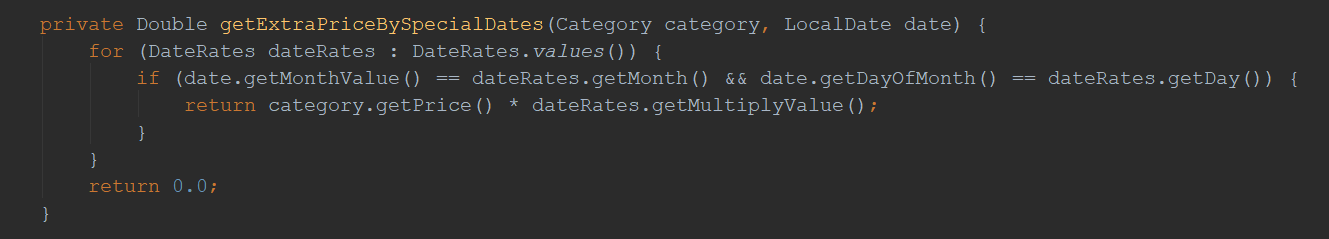


Figura 32: Metoda care dă un adaos de preț în caz ca ziua primită este pe o listă cu zile în care se majorează prețul

Pentru aflarea noului procentaj al prețului în funcție de numărul de zile al rezervării, este folosit un nou enum a cărui câmpuri sunt un număr de zile și un procentaj de reducere, iar dacă numărul de zile este mai mare sau egal ca cel din enum vom returna valoarea 1 minus procentajul corespunzator acelui număr de zile, astfel rezultând noul procentaj al prețului.

Exemplu: dacă in enum se găsește valoarea 30 pentru numărul de zile și procentajul de reducere 0.1, în cazul în care metoda este apelată pentru o rezervare de 35 de zile metoda returnează valoarea 0.9 pentru a o înmulți cu prețul pentru rezervare calculat inițial, astfel oferind o reducere de 10% la acel preț.

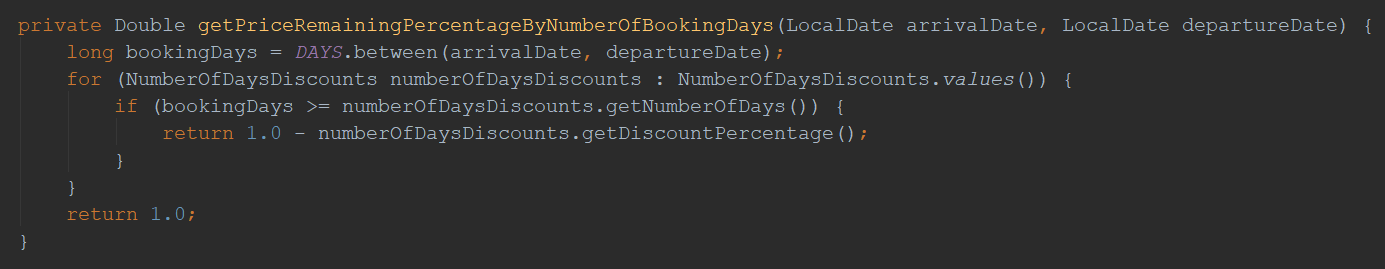


Figura 33: Metoda care oferă o reducere a prețului în funcție de numărul de zile pentru care este facută rezervarea prin oferirea unui procentaj pentru a se înmulți cu valoarea actuală a prețului

La apelul metodei getPriceIncreaseByNumberOfBookingInLastDay se va face un apel la metoda getNumberOfBookingsIn24HoursIntervalBeforeNow pentru a afla numărul de rezervări făcute în ultimele 24 de ore, iar dacă numărul de rezervări este îndeajuns de mare se va crește procentajul prețului după cum se vede în figura de mai jos.

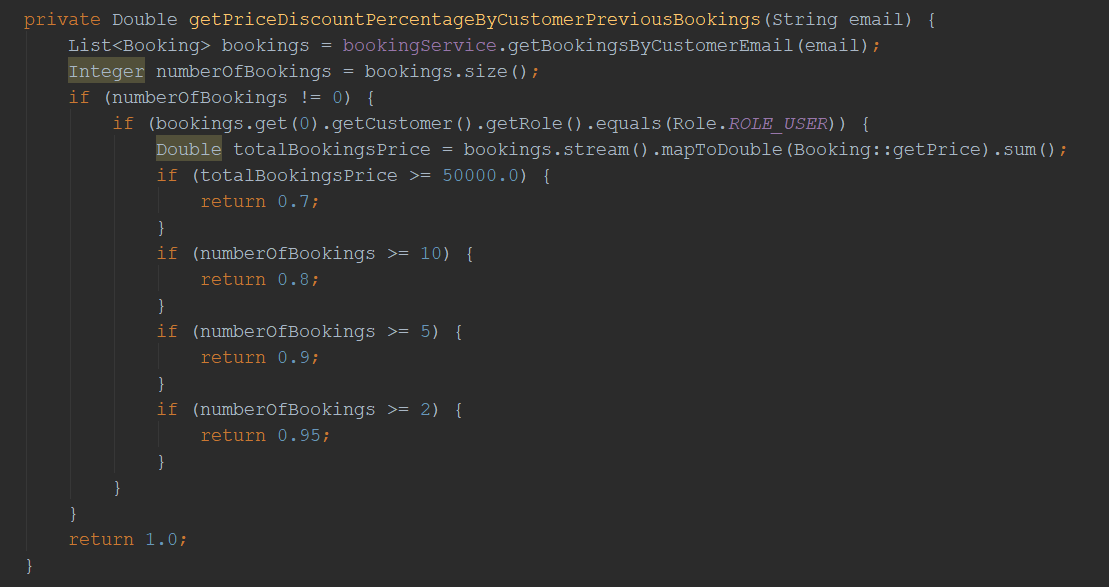


Figura 34: Metoda care mărește procentajul prețului actual dacă au fost făcute multe rezervări în ultimele 24 de ore

În final, la apelarea metodei pentru calcularea noului pocentaj în funcție de rezervarile precedente ale utilizatorului (vezi figura de mai jos), se va face un apel la o metodă din serviciul bookingService pentru a primi o listă cu rezervările precedente ale utilizatorului în funcție de email. Mai departe, în cazul în care lista nu este goală și utilizatorul este unul cu rol ROL\_USER (adică client normal) se va returna noul procentaj al prețului în funcție de numărul total de rezervări și de prețul total al rezervărilor precedente.

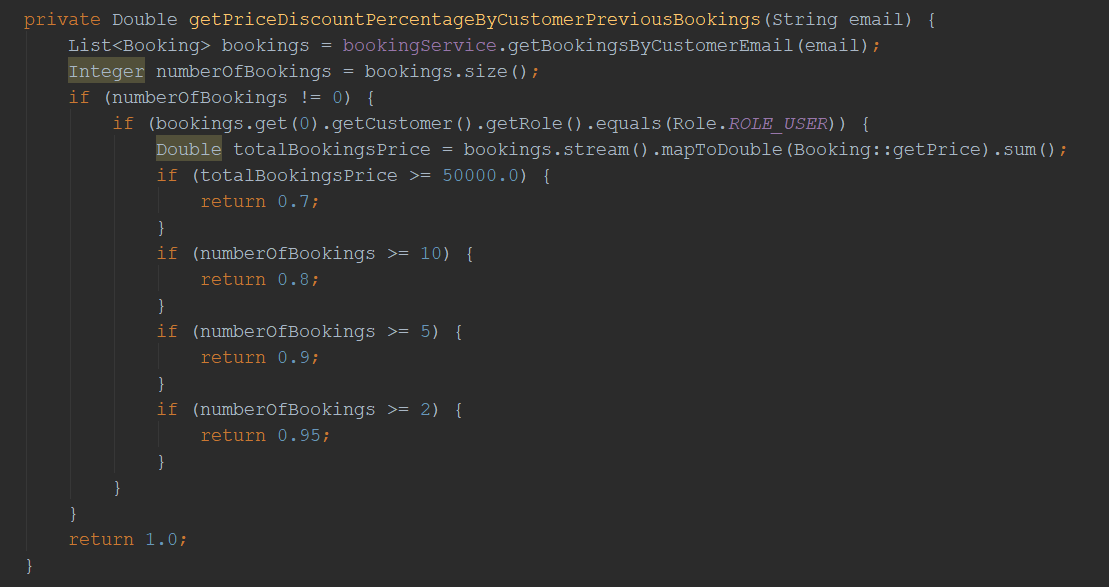


Figura 35: Metoda care oferă un procentaj scăzut al prețului în funcție de rezervările anterioare ale clientului

**3.6 Serviciu emailuri**

La trimiterea de emailuri se folosește Gmail SMTP. Pentru a se putea folosi acest serviciu trebuie adăugată dependința spring-boot-starter-mail și trebuie făcute configurările pentru serverul care trimite emailuri (configurările se fac în fișierul application.properties după cum se vede în figura de mai jos).

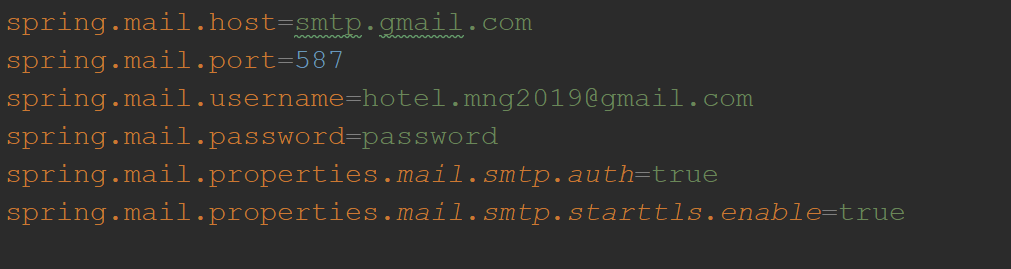


Figura 36: Configurări server pentru mail (gmail)

Odată ce avem aceste configurări făcute pentru a trimite un email trebuie creat mesajul, în care punem subiectul, destinatarul și mesajul propriu (vezi tabelul 25). De asemenea, la trimiterea unui email vom crea un nou fir de execuție care se va ocupa cu trimiterea acestuia deoarece trimiterea unui email poate dura o perioadă de timp relativ lungă (mai mult de 2-3 secunde) și nu vrem ca aplicația noastră să fie blocată în acest timp.

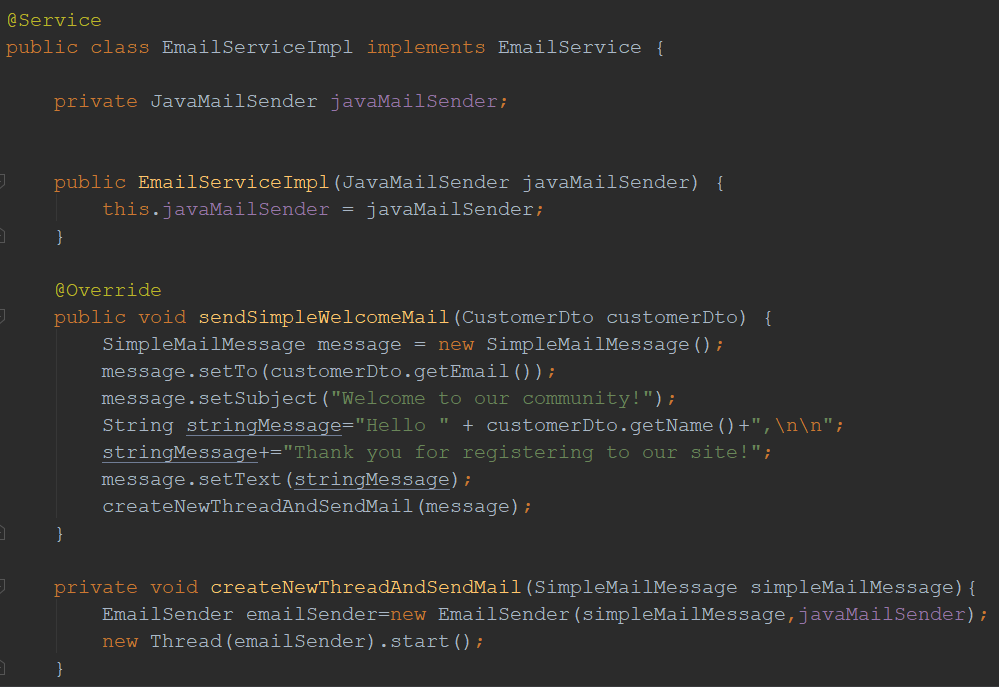


Figura 37: Clasă creare și trimitere email

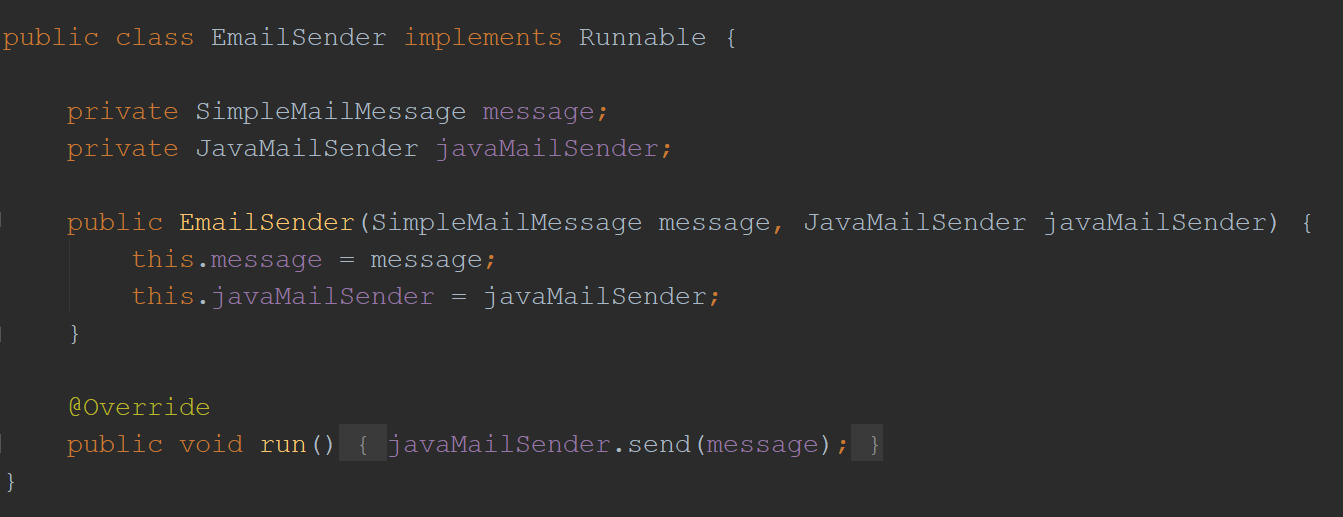


Figura 38: Clasă folosită pentru trimiterea unui email într-un nou fir de execuție

**3.7 Plată PayPal**

La finalizarea rezervării unei camere utilizator trebuie să efectueze plata rezervării. Această plată este facut prin PayPal în mediul de testare oferit de PayPal numit “sandbox”. Realizarea plății PayPal în mediul sandbox este similară cu plata reală, pe parte de cod schimbându-se doar un parametru al unei funcții care semnalează daca plata să fie făcută în sistemul real sau în sistemului de test. Motivul folosirii acestui sistem de test în aplicație pentru moment este faptul că este gratis și este modalitatea recomandată pentru a dezvolta o aplicație, la final putând trece într-un mediu real foarte ușor cu o investiție de bani pentru a îți deschide conturile necesare.

Pentru a începe dezvolatrea sitemului de plată prin PayPal inițial trebuie să se creeze un cont la adresa [https://developer.paypal.com](https://developer.paypal.com/). Mai departe vom crea cel puțin două conturi PayPal de testare (conturi sandbox), unul din aceste conturi fiind folosit ca și contul principal pe care se vor primi bani (contul hotelului), și al doilea cont fiind folosit ca și contul care va trimite banii (contul clientului). Aceste conturi se pot configura cu orice sumă de bani, parolă și adresă de email.

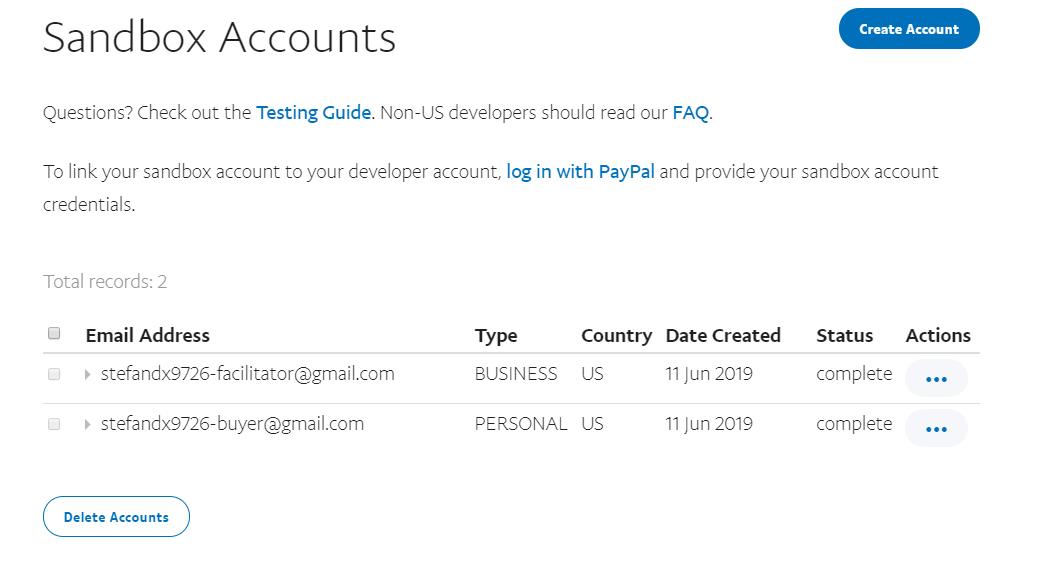


Figura 39: Creare conturi PayPal Sandbox

Următorul pas este introducerea sitemului extern pentru dezvolatrea unui sistem de plată PayPal. Pentru folosirea acestui sistem este folosit un SDK oferit de PayPal pentru dezvoltarea de aplicații (<https://github.com/paypal/PayPal-Java-SDK>). Apoi vom folosi aceast SDK pentru a crea un obiect de tip plată (vezi figura de mai jos) care conține informațiile necesare pentru efectuarea unei plăți, adică, printre altele, metoda de plată, id-ul și secretul contului care ar urma să primeașcă banii, suma și tipul monezii de plată.

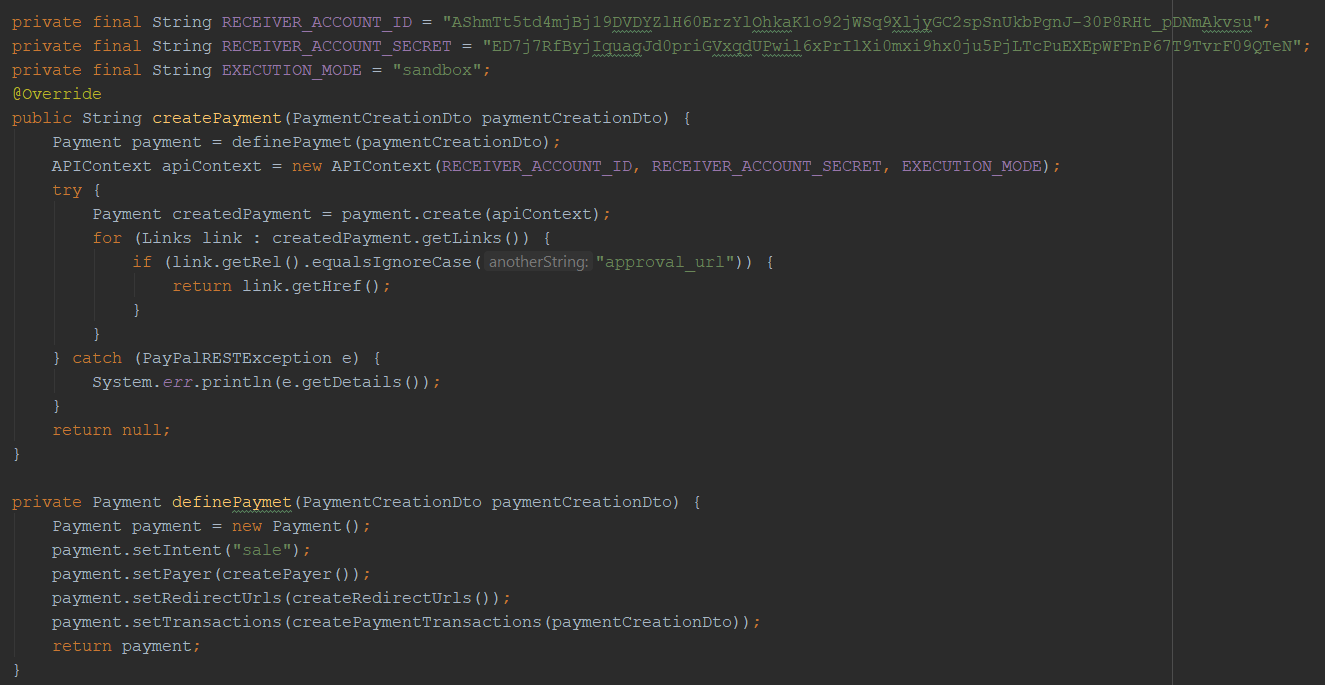


Figura 40: Metodă creare plată

Pentru efectuarea unei plățit reale trebuie schimbată valoarea constatntei EXECUTION\_MODE în “live”.

Odată apleată metoda createPayment din figura de mai sus, obiectul de tip plată cu informațiile trimise de noi este creat și este un returnat un link care va duce la o pagină de logare în contul de PayPal (în cazul nostru la contul de tip sandbox) cum se vede în figura de mai jos. Aici utilizatorul va trebui să-și introducă informațiile și dupa autentificare să decidă daca aceptă sau nu plata și în ce mod să se facă plata (vezi figura 42).

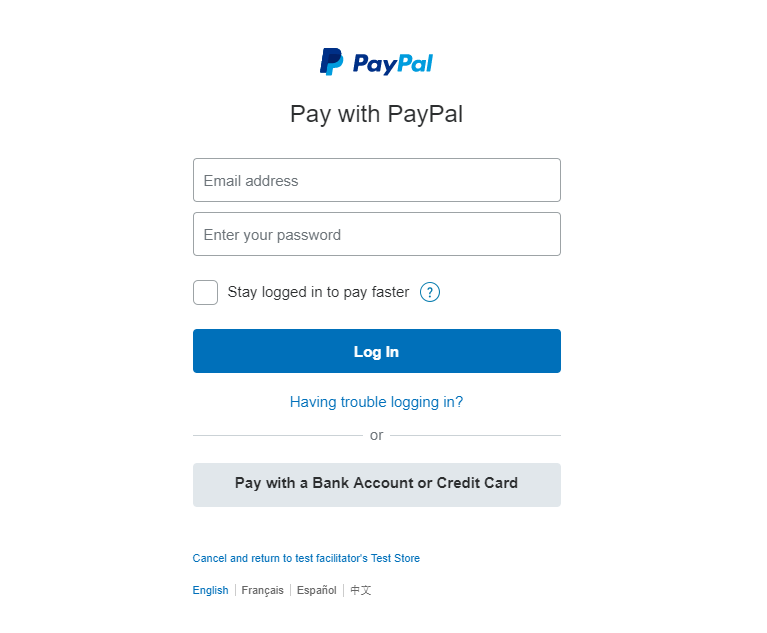


Figura 41: Pagină autenificare PayPal

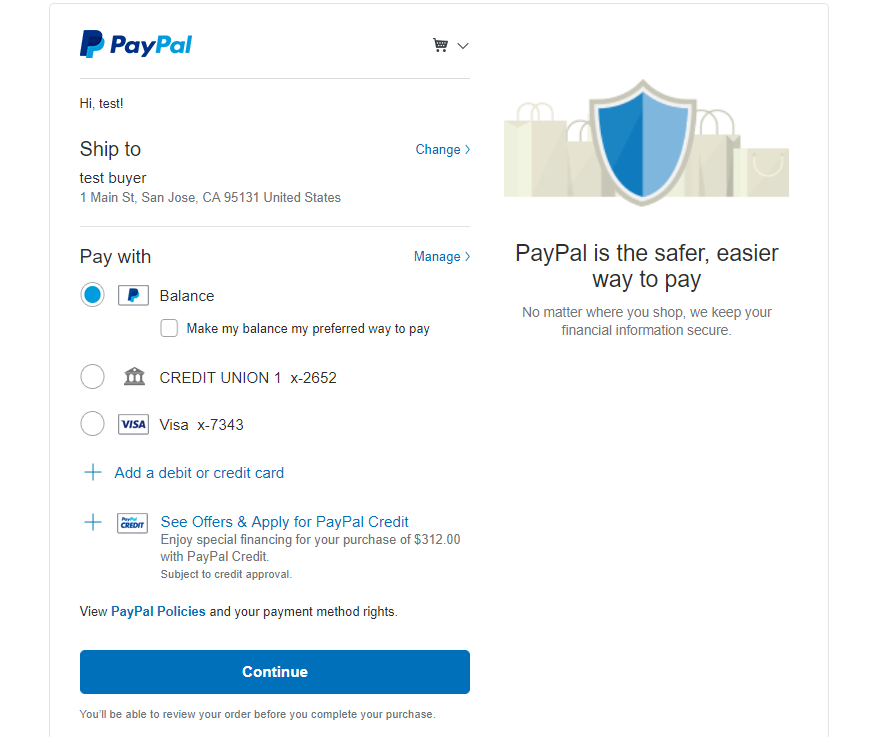


Figura 42: Slectare metodă plată și confirmare

Dacă utilizatorul s-a autentificat și a ales să accepte plata acesta va fi redirecționat spre adresa de confirmare la care în parametri URL-ului se vor adăuga id-ul plății și id-ul utilizatorului care efectuează plata. În caz contrar, acesta va fi redirecționat către adresa folosită la anularea unei plăți. În final, dacă a fost acceptată plata se vor folosi acei parametri pentru executarea plății după cum se vede în figura de mai jos.

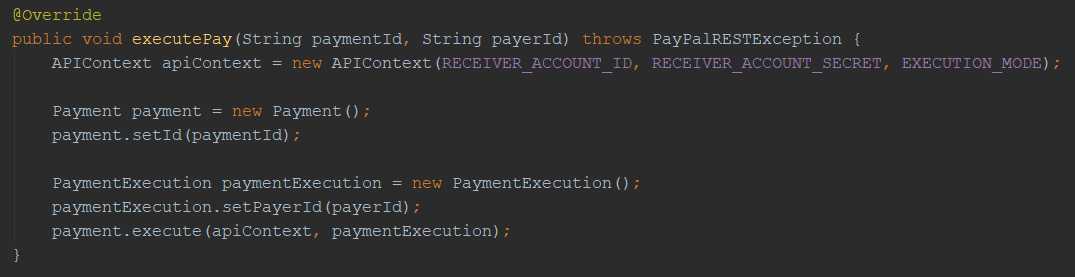


Figura 43: Metodă care execută operația de plată

Concluzii

Așadar, această aplicație, deși pare asemănătoare cu alte aplicații de pe piața, oferă un serviciu foarte diferit clienților, adică oferă posibilitatea cazării într-o cameră și comenzilor de room-service doar prin intermediul aplicației fața de restul aplicațiilor existente la care poți face rezervarea online prin intermediul aplicației dar apoi pentru a intra în camera de hotel este necesar să mergi la recepție pentru a face check-in-ul. Acest lucru consider că este un mare plus doarece nu poate apărea riscul pierderii unei perioade lungi de timp la recepția hotelului și, de asemenea, consider că este în ton cu felul în care se dezvoltă societatea în ziua de azi, adică să faciliteze comoditatea clientului care vrea să se cazeze la hotel.

Aplicație este facută să fie responsivă pentru orice tip de dispozitiv și respectă tehnicile de clean code fiind ușor de dezvoltat mai departe pentru a adăuga noi funcționalități.

Pe viitor, această aplicație ar putea fi extinsă prin adăugarea posibilității de a face rezervări pentru anumite săli de conferința alea hotelului și a face rezervări la evenimente de tip spa și masaj. La algoritmul folosit pentru crearea unui preț variabil s-ar putea adauga înca un criteriu pentru calcularea prețului prin preluarea informațiilor legate de prețul camerelor de pe alte aplicații ale unor hoteluri existente din zona hotelului care ar folosi aplicația.

De asemenea, s-ar putea adăuga un mecansim prin care angajați să adauge în aplicație toate produsele consumate, de exemplu prosoape, șampon, săpun. Apoi pe baza istoricului acestora și pe baza istoricului produselor consumate în comenzile de room-service s-ar putea folosi un algoritm de învățare automată pentru a prezice stocul necesar pe viitor al hotelului pentru acest tip de produse.

**Bibliografie**

1. <https://spring.io/guides/gs/rest-service/>

2. <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/>

3. <https://www.baeldung.com/spring-controller-vs-restcontroller>

4. <https://www.baeldung.com/spring-requestmapping>

5. <https://www.baeldung.com/spring-data-jpa-query>

6. <https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring>

7. <https://dzone.com/articles/bounty-spring-boot-and-postgresql-database>

8. <https://www.vogella.com/tutorials/JUnit/article.html>

9. <https://www.vogella.com/tutorials/Mockito/article.html>

10. <https://www.baeldung.com/spring-email>

11. <https://angular.io/tutorial>

12. <https://coursetro.com/posts/code/171/Angular-7-Tutorial---Learn-Angular-7-by-Example>

13. <https://material.angular.io/>

14. <https://www.jetbrains.com/help/idea/running-and-debugging-typescript.html>

15. <https://getbootstrap.com/>

16. <https://www.w3schools.com/bootstrap4/default.asp>

17. <https://www.primefaces.org/primeng/#/>

18. <https://angular.io/guide/styleguide>

19. <https://itnext.io/choosing-a-highly-scalable-folder-structure-in-angular-d987de65ec7>

20. <https://github.com/mathisGarberg/angular-folder-structure>

21. <https://malcoded.com/posts/angular-fundamentals-modules/>

22. <https://www.technouz.com/4772/angular-6-app-structure-with-multiple-modules/>

23. <https://code-maze.com/angular-best-practices/>

24. <https://www.freecodecamp.org/news/how-to-center-things-with-style-in-css-dc87b7542689/>

25. <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

26. <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Ordering_Flex_Items>

27. <https://flexboxfroggy.com/>

28. <https://www.baeldung.com/security-spring>

29. <https://www.baeldung.com/securing-a-restful-web-service-with-spring-security>

30. <https://www.baeldung.com/spring-security-oauth-jwt>

31. <https://chariotsolutions.com/blog/post/angular-2-spring-boot-jwt-cors_part2/>

32. <https://mherman.org/blog/token-based-authentication-with-angular/#localstorage>

33. <https://codecraft.tv/courses/angular/dependency-injection-and-providers/tokens/>

34. <https://medium.com/engineerbabu/angular-authentication-using-jwt-d846c5ce0ac6>

35. <https://medium.com/@ryanchenkie_40935/angular-authentication-using-route-guards-bf7a4ca13ae3>

36. <https://nirajsonawane.github.io/2018/10/27/Angular-Material-Tabs-with-Router/>

37. <https://medium.com/@nikhildevre/creating-tabs-using-angular-material-2-and-angular-4-routing-3634c3d0f7cc>

38. <https://blog.angularindepth.com/angular-router-series-secondary-outlets-primer-139206595e2>

39. <https://www.w3schools.com/howto/howto_js_tabs.asp>

40. <https://css-tricks.com/left-and-right/>

41. <https://css-tricks.com/couple-takes-sticky-footer/>

42. <https://github.com/paypal/PayPal-Java-SDK>

43. <https://developer.paypal.com/docs/api/quickstart/payments/#additional-information>

44. <https://developer.paypal.com/developer/accountStatus/>

45. <https://c4model.com/>

46. <https://structurizr.com/express?src=/static/express/bigbankplc/system-context.json>

47. <https://www.draw.io/>

48. <https://hatchful.shopify.com/>

49. <https://www.pexels.com/search/hotel/>

50. <https://pixabay.com/images/search/hotel/>

51. <https://unsplash.com/search/photos/room-service>