



FIVE HOTELS

- AFRA TAÍZA
- HELZA ARAGÃO
- MAURO MARTINS
- PATRÍCIA DELVEQUI
- DARLAN MENEZES

DIVISÃO DO TIME

Front-end

Afra Taíza e Helza Aragão

Testes

Mauro Martins

Back-end

Patrícia Delvequi e Darlan Menezes

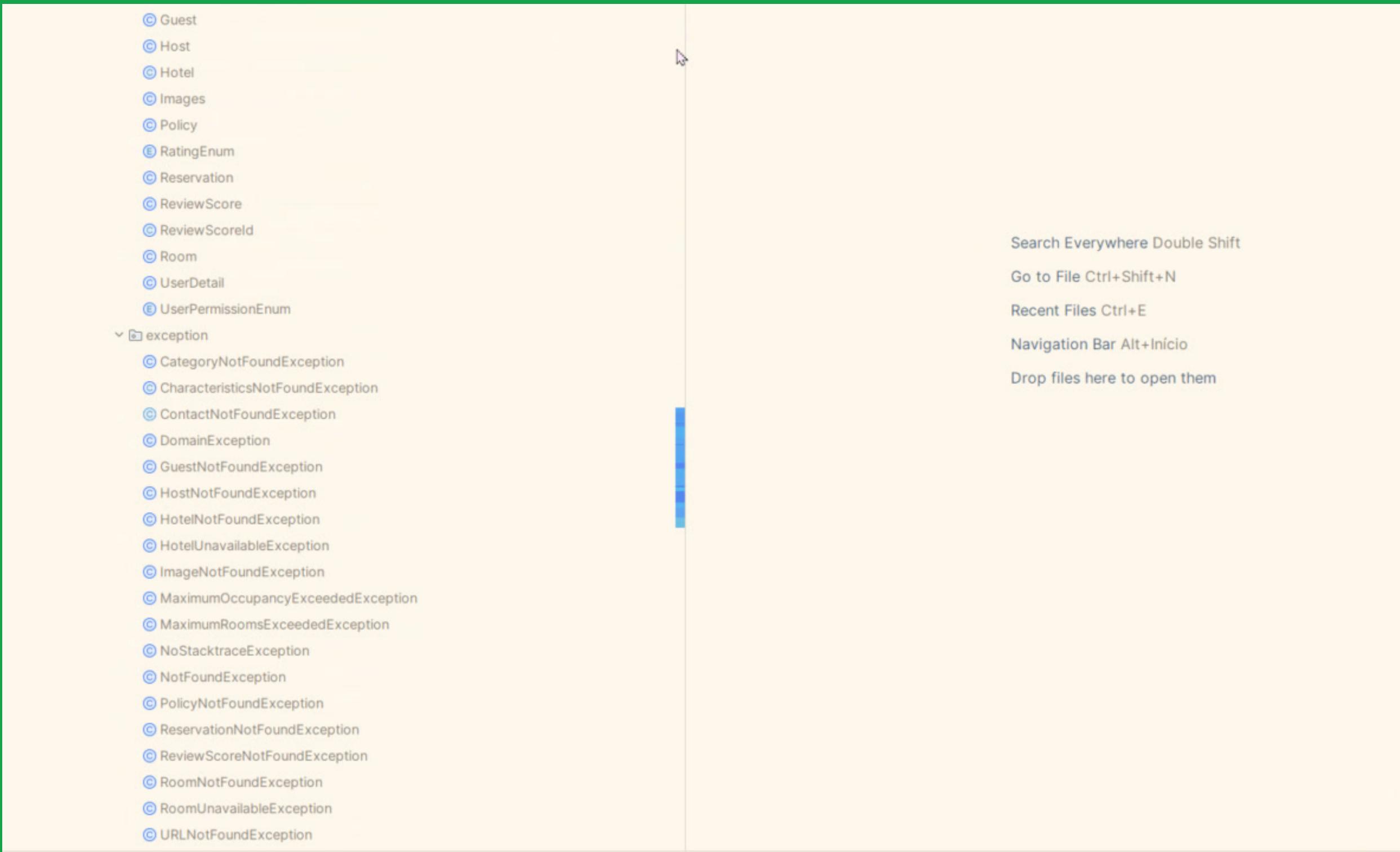


ESCOLHA DA CATEGORIA DO PROJETO

Hotéis, hotels, pousadas e apartamentos

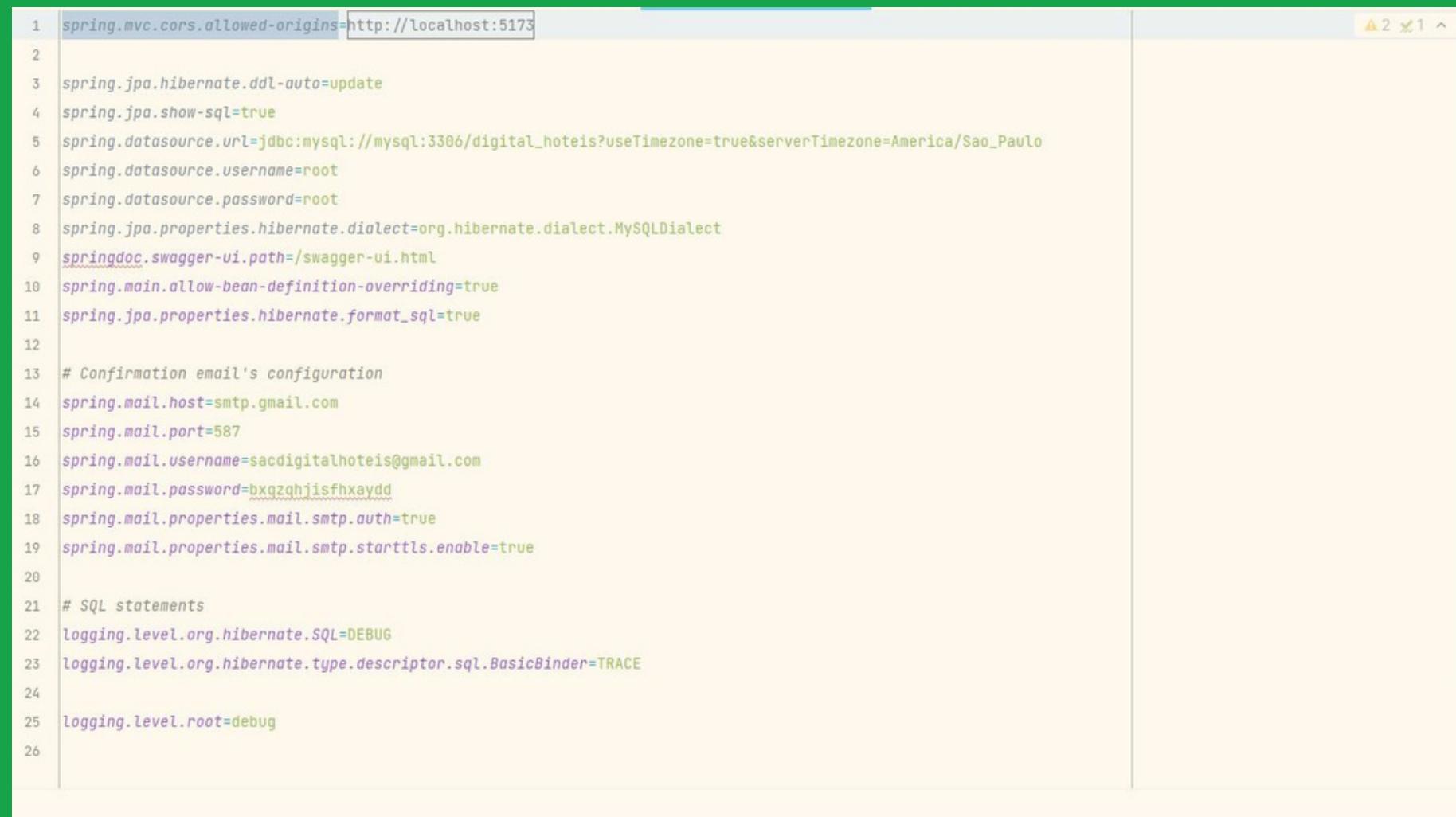
BACK-END

DETALHES DO QUE FEITO



BACK-END

DETALHES DO QUE FEITO



A screenshot of a code editor showing the `application.properties` file. The file contains configuration properties for a Spring-based application, including database settings, email configuration, and logging levels.

```
1 spring.mvc.cors.allowed-origins=http://localhost:5173
2
3 spring.jpa.hibernate.ddl-auto=update
4 spring.jpa.show-sql=true
5 spring.datasource.url=jdbc:mysql://mysql:3306/digital_hoteis?useTimezone=true&serverTimezone=America/Sao_Paulo
6 spring.datasource.username=root
7 spring.datasource.password=root
8 spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
9 springdoc.swagger-ui.path=/swagger-ui.html
10 spring.main.allow-bean-definition-overriding=true
11 spring.jpa.properties.hibernate.format_sql=true
12
13 # Confirmation email's configuration
14 spring.mail.host=smtp.gmail.com
15 spring.mail.port=587
16 spring.mail.username=sacdigitalhoteis@gmail.com
17 spring.mail.password=bxqzqhijsfhxaydd
18 spring.mail.properties.mail.smtp.auth=true
19 spring.mail.properties.mail.smtp.starttls.enable=true
20
21 # SQL statements
22 logging.level.org.hibernate.SQL=DEBUG
23 logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
24
25 logging.level.root=debug
26
```

Configuração da `application.properties`

Foram adicionadas as configurações necessárias para o Hibernate atualizar o banco de dados, sua conexão com o container do Docker, assim como senhas, caminho da url do swagger e algumas funcionalidades extras.

O servidor de email SMTP que foi feito através de gmail também foi configurado nesse arquivo.

E para facilitar o processo de criação e solução de problemas relacionados ao código foi habilitado o nível de DEBUG do hibernate, isso ajudou muito, muito mesmo.

BACK-END

DETALHES DO QUE FEITO

```
① AuthenticationController.java ×
75     log.error("Error during sign-up: Bad request - {}", e.getMessage(), e);
76     return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
77 } catch (HttpClientErrorException e) {
78     if (e.getStatusCode() == HttpStatus.CONFLICT) {
79         log.error("Error during sign-up: Conflict - Email already in use. Email: {}", request.getEmail(), e);
80         return ResponseEntity.status(HttpStatus.CONFLICT)
81             .body("Infelizmente não foi possível registrar. Por favor, tente novamente mais tarde");
82     } else {
83         log.error("Error during sign-up: HTTP Status - {}. Email: {}", e.getStatusCode(), request.getEmail(), e);
84         return ResponseEntity.status(e.getStatusCode()).body("Ocorreu um erro: " + e.getMessage());
85     }
86 } catch (Exception e) {
87     log.error("Error during sign-up: Internal Server Error. Email: {}", request.getEmail(), e);
88     return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Erro interno do servidor: " + e.getMessage());
89 }
90 }
91
no usages
92 @RequestMapping(value = "/confirm-account", method = {RequestMethod.GET, RequestMethod.POST})
93 public ResponseEntity<?> confirmUserAccount(@RequestParam("token") String confirmationToken) {
94     ConfirmationToken token = confirmationTokenRepository.findByConfirmationToken(confirmationToken);
95
96     if (token != null) {
97         UserDetail user = token.getUserEntity();
98
99         if (!user.isEnabled()) {
100             user.setEnabled(true);
101             String jwt = String.valueOf(authenticationService.signIn(new SignIn(user.getEmail(), password: null), String.valueOf(token)));
102
103             return ResponseEntity.ok().body("Email successfully verified!");
104         } else {
105             return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("User is already confirmed.");
106         }
107     } else {
108         return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid or expired token.");
109     }
110 }
```

Controller de autenticação e a classe de implementação do serviço de autenticação

BACK-END

DETALHES DO QUE FEITO

```
① AuthenticationServiceImpl.java ×
43 ④ @
44     public String signUp(SignUp request) {
45         if (!request.getEmail().equals(request.getRtype_email())) {
46             throw new IllegalArgumentException("Error: The email and rtype_email must match.");
47         }
48         UserDetail user = UserDetail.builder()
49             .name(request.getName())
50             .surname(request.getSurname())
51             .email(request.getEmail())
52             .password(passwordEncoder.encode(request.getPassword()))
53             .isHost(request.isHost())
54             .build();
55         if (userRepository.existsByEmail(user.getEmail())) {
56             throw new IllegalArgumentException("Erro: O e-mail já está em uso");
57         }
58         if (user.isHost()) {
59             userRepository.saveHost(user.getEmail(), UserPermissionEnum.ADMIN, user.getName(), user.getSurname());
60         } else {
61             userRepository.saveGuest(user.getEmail(), UserPermissionEnum.USER, user.getName(), user.getSurname());
62         }
63         ConfirmationToken confirmationToken = new ConfirmationToken(user);
64         confirmationTokenRepository.save(confirmationToken);
65
65         SimpleMailMessage mailMessage = new SimpleMailMessage();
66         mailMessage.setTo(user.getEmail());
67         mailMessage.setSubject("Conclua seu cadastro");
68         mailMessage.setText("Para confirmar sua conta, clique no link por favor: "
69             + "<a href='http://localhost:9090/v1/authentication/confirm-account?token="
70             + confirmationToken.getConfirmationToken() + "'>Clique aqui</a>");
71         emailService.sendEmail(mailMessage);
72
73         return confirmationToken.getConfirmationToken();
74     }
75
76     no usages
77 ④ @Override
78     public ResponseEntity<?> confirmUserAccount(String confirmationToken) {
79         ConfirmationToken token = confirmationTokenRepository.findByConfirmationToken(confirmationToken);
80
81         if (token != null && token.isConfirmed()) {
82             return ResponseEntity.ok("A conta já está confirmada.");
83         }
84         token.setConfirmed(true);
85         confirmationTokenRepository.save(token);
86
87         return ResponseEntity.ok("A conta foi confirmada com sucesso!");
88     }
89
89     no usages
```

Controller de autenticação e a classe de implementação do serviço de autenticação

BACK-END

DETALHES DO QUE FEITO

```
② AuthenticationServiceImpl.java ×
1  package br.com.zup.edu.authentication;
2
3  import br.com.zup.edu.model.UserDetail;
4  import br.com.zup.edu.model.UserPermissionEnum;
5  import br.com.zup.edu.repository.UserRepository;
6  import br.com.zup.edu.service.ConfirmationToken;
7  import br.com.zup.edu.service.ConfirmationTokenRepository;
8  import br.com.zup.edu.service.EmailService;
9  import br.com.zup.edu.util.PasswordEncoder;
10 import org.springframework.beans.factory.annotation.Autowired;
11 import org.springframework.stereotype.Service;
12 import org.springframework.transaction.annotation.Transactional;
13
14 @Service
15 @Transactional
16 public class AuthenticationServiceImpl implements AuthenticationService {
17
18     @Autowired
19     private UserRepository userRepository;
20
21     @Autowired
22     private ConfirmationTokenRepository confirmationTokenRepository;
23
24     @Autowired
25     private EmailService emailService;
26
27     @Autowired
28     private PasswordEncoder passwordEncoder;
29
30
31     public String signUp(SignUp request) {
32
33         if (!request.getEmail().equals(request.getRetype_email())) {
34             throw new IllegalArgumentException("Error: The email and retype_email must match.");
35         }
36
37         UserDetail user = UserDetail.builder()
38             .name(request.getName())
39             .surname(request.getSurname())
40             .email(request.getEmail())
41             .password(passwordEncoder.encode(request.getPassword()))
42             .isHost(request.isHost())
43             .build();
44
45         if (userRepository.existsByEmail(user.getEmail())) {
46             throw new IllegalArgumentException("Erro: O e-mail já está em uso");
47         }
48
49         if (user.isHost()) {
50             userRepository.saveHost(user.getEmail(), UserPermissionEnum.ADMIN, user.getName(), user.getSurname());
51         } else {
52             userRepository.saveGuest(user.getEmail(), UserPermissionEnum.USER, user.getName(), user.getSurname());
53         }
54
55         ConfirmationToken confirmationToken = new ConfirmationToken(user);
56         confirmationTokenRepository.save(confirmationToken);
57
58         SimpleMailMessage mailMessage = new SimpleMailMessage();
59         mailMessage.setTo(user.getEmail());
60         mailMessage.setSubject("Conclua seu cadastro");
61         mailMessage.setText("Para confirmar sua conta, clique no link por favor: "
62             + "<a href='http://localhost:9090/v1/authentication/confirm-account?token=" +
63             confirmationToken.getConfirmationToken() + "'>Clique aqui</a>");
64         emailService.sendEmail(mailMessage);
65
66         return confirmationToken.getConfirmationToken();
67     }
68
69     no usages
70
71     @Override
72     public ResponseEntity<?> confirmUserAccount(String confirmationToken) {
73
74         ConfirmationToken token = confirmationTokenRepository.findByConfirmationToken(confirmationToken);
75
76         if (token != null) {
77             token.setConfirmed(true);
78             confirmationTokenRepository.save(token);
79
80             return ResponseEntity.ok("User account confirmed successfully!");
81         } else {
82             return ResponseEntity.badRequest().body("Confirmation token not found or already confirmed");
83         }
84     }
85 }
```

Controller de autenticação e a classe de implementação do serviço de autenticação

BACK-END

DETALHES DO QUE FEITO

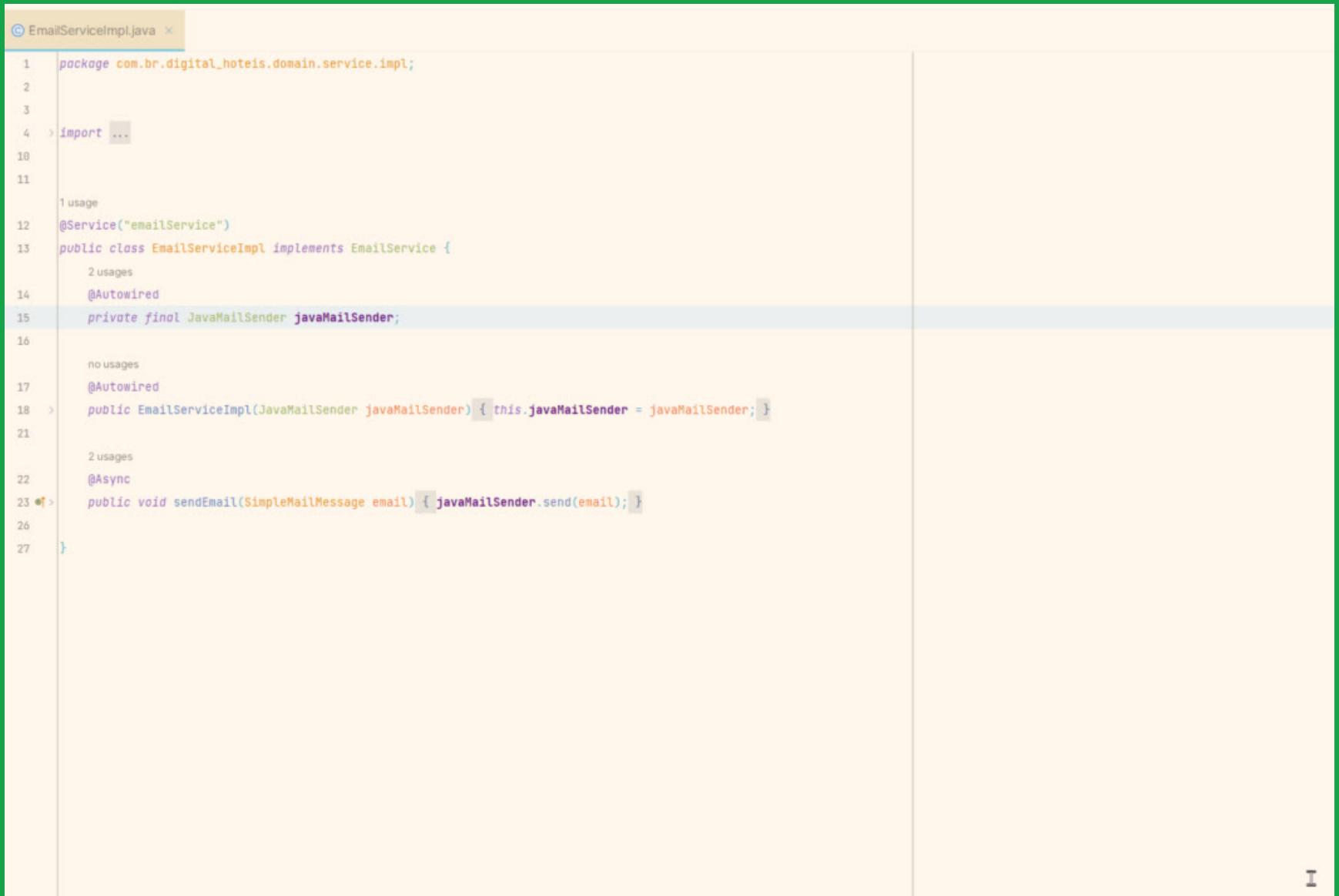
```
UserServiceImpl.java x

38     if (host.getEmail() != null) {
39         Optional<UserDetail> existingHost = userRepository.findByEmailAndIsHost(host.getEmail(), isHost: true);
40
41         if (existingHost.isPresent()) {
42             return ResponseEntity.badRequest().body("Error: Host with the same email already exists!");
43         }
44
45         host.setName(host.getName());
46         host.setSurname(host.getSurname());
47         host.setRole(UserPermissionEnum.ADMIN);
48         return saveUser(host);
49     } else {
50
51         log.error("Email address is null for host: {}", host);
52         return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
53             .body("Internal error: Email address is null");
54     }
55 }
56
57 no usages
58 @Override
59
60 public ResponseEntity<?> saveGuest(UserDetail guest) {
61
62     if (guest.getEmail() != null) {
63         Optional<UserDetail> existingGuest = userRepository.findByEmailAndIsHost(guest.getEmail(), isHost: false);
64
65         if (existingGuest.isPresent()) {
66             return ResponseEntity.badRequest().body("Error: Guest with the same email already exists!");
67         }
68
69         guest.setName(guest.getName());
70         guest.setSurname(guest.getSurname());
71         guest.setRole(UserPermissionEnum.USER);
72         return saveUser(guest);
73     } else {
74
75     }
76 }
```

Controller de autenticação e a classe de implementação do serviço de autenticação

BACK-END

DETALHES DO QUE FEITO



```
⑤ EmailServiceImpl.java ×
1 package com.br.digital_hoteis.domain.service.impl;
2
3
4 > import ...
10
11
1 usage
12 @Service("emailService")
13 public class EmailServiceImpl implements EmailService {
14     2 usages
15     @Autowired
16     private final JavaMailSender javaMailSender;
17
18     no usages
19     @Autowired
20     public EmailServiceImpl(JavaMailSender javaMailSender) { this.javaMailSender = javaMailSender; }
21
22     2 usages
23     @Async
24     public void sendEmail(SimpleMailMessage email) { javaMailSender.send(email); }
25
26
27 }
```

JavaMailSender:
Classe que auxilia no envio do email

BACK-END

DETALHES DO QUE FEITO

```
⑤ WebServerSecurityConfiguration.java ×
31     private final UserService userService;
32
33     4 usages
34     private static final String[] AUTH_WHITELIST = {
35         "/api/v1/auth/**",
36         "/v3/api-docs/**",
37         "/v3/api-docs.yaml",
38         "/swagger-ui/**",
39         "/swagger-ui.html",
40         "/Categories/**",
41         "/v1/authentication/confirm-account",
42         "/v1/authentication/**"
43     };
44
45     no usages
46     @Override
47     public void addCorsMappings(CorsRegistry registry) {
48         registry.addMapping(pathPattern: "/**")
49             .allowedOrigins("http://localhost:5173")
50             .allowedMethods("GET", "POST", "PUT", "DELETE")
51             .allowedHeaders("*")
52             .allowCredentials(true)
53             .maxAge(3600);
54
55     no usages
56     @Bean
57     public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
58         http.csrf(AbstractHttpConfigurer::disable)
59             .authorizeHttpRequests(request →
60                 request
61                     .requestMatchers(HttpMethod.POST, ...patterns: "/v1/authentication/**").permitAll()
62                     .requestMatchers(HttpMethod.POST, AUTH_WHITELIST).permitAll()
63                     .requestMatchers(HttpMethod.GET, AUTH_WHITELIST).permitAll()
64                     .requestMatchers(HttpMethod.PUT, AUTH_WHITELIST).permitAll()
65                     .requestMatchers(HttpMethod.DELETE, AUTH_WHITELIST).permitAll()
66                     .anyRequest().authenticated())
67     }
68 }
```

Considerei como desafio a configuração da classe “WebServerSecurityConfiguration” , depois de tentativas e diversos testes ela foi ajustada.

BACK-END

DETALHES DO QUE FEITO

```
① HotelApi.java ×
40  @GetMapping("{hotelId}/hosts")
41  @PreAuthorize("hasAuthority('ADMIN')")
42  @ResponseEntity<Page<HostSummaryResponse>> findHostsByHotelId(@PathVariable UUID hotel_id,
43  @PageableDefault Pageable pageable);
44  |
45  no usages 1 implementation
46  @GetMapping("{hotelId}/guests")
47  @PreAuthorize("hasAuthority('ADMIN')")
48  @ResponseEntity<Page<GuestSummaryResponse>> findGuestByHotelId(@PathVariable UUID hotel_id,
49  @PageableDefault Pageable pageable);
50  |
51  no usages 1 implementation
52  @GetMapping("{hotelId}/policy")
53  @PreAuthorize("hasAuthority('USER')")
54  @ApiResponse(responseCode = "200", description = "Retrieve hotel with policy",
55  content = @Content(mediaType = MediaType.APPLICATION_JSON_VALUE,
56  schema = @Schema(implementation = HotelDetailedResponse.class)))
57  @ResponseEntity<Map<String, Object>> findHotelWithPolicy(@PathVariable UUID hotel_id);
58  |
59  no usages 1 implementation
60  @PostMapping("{hotelId}/reviews")
61  @PreAuthorize("hasAuthority('USER')")
62  @ResponseEntity<Void> addReviewScore(@PathVariable UUID hotel_id,
63  @RequestParam UUID user_id,
64  @RequestParam Integer review_scores);
65  |
66  no usages 1 implementation
67  @PostMapping
68  @PreAuthorize("hasAuthority('ADMIN')")
69  @ResponseEntity<Void> createHotel(@RequestBody @Valid CreateHotelRequest request);
70  |
71  no usages 1 implementation
72  @PatchMapping("{hotelId}")
73  @ResponseEntity<Void> updateHotel(@PathVariable UUID hotel_id,
74  @RequestBody Map<String, Object> fields);
75  |
76  no usages 1 implementation
77  @PostMapping("{hotelId}/policies")
78  @ResponseEntity<Void> addPoliciesToAHotelById(@PathVariable UUID hotelId, @RequestBody @Valid CreatePolicyRequest request);
```

Foi desafiador criar o controller de Reservas e de Hotel devido aos detalhes que cada classe tinha, foram necessários constantes ajustes.

BACK-END

DETALHES DO QUE FEITO

```
© HotelServiceImpl.java ×

85
86
87     1 usage
88     @Override
89     public void addReviewScore(UUID hotel_id, UUID user_id, Integer review_scores) {
90         if (review_scores < 1 || review_scores > 5) {
91             throw new IllegalArgumentException("Rating must be between 1 and 5.");
92         }
93
94         Hotel hotel = hotelRepository.findById(hotel_id)
95             .orElseThrow(() -> new HotelNotFoundException(hotel_id));
96
97         if (hotel.getReview_scores() == null) {
98             hotel.setReview_scores(new HashSet<>());
99         }
100
101         ReviewScore reviewScore = ReviewScore.newReviewScore(hotel_id, user_id, review_scores);
102
103         reviewScoreService.setHotelForReviewScore(hotel, reviewScore);
104
105         hotelRepository.save(hotel);
106     }

107     1 usage
108     public void addCharacteristics(UUID hotel_id, Characteristics characteristics) {
109         Hotel hotel = hotelRepository.findHotelByIdWithCharacteristics(hotel_id);
110
111         if (hotel != null) {
112             hotel.getCharacteristics().add(characteristics);
113             hotelRepository.save(hotel);
114         } else {
115             hotelRepository.findById(hotel_id)
116                 .orElseThrow(() -> new HotelNotFoundException(hotel_id));
117         }
118     }

119     1 usage
```

BACK-END

DETALHES DO QUE FEITO

```
© HotelController.java ×

214     hotel.getCity().getName(),
215     hotel.getCity().getStreet(),
216     hotel.getCity().getDistrict(),
217     hotel.getCity().getState(),
218     hotel.getCity().getZipcode(),
219     hotel.getCity().getCountry(),
220     hosts,
221     policyResponse,
222     hotel.getMax_number_of_rooms(),
223     images
224 };
225 return ResponseEntity.ok(response);
226 }

227

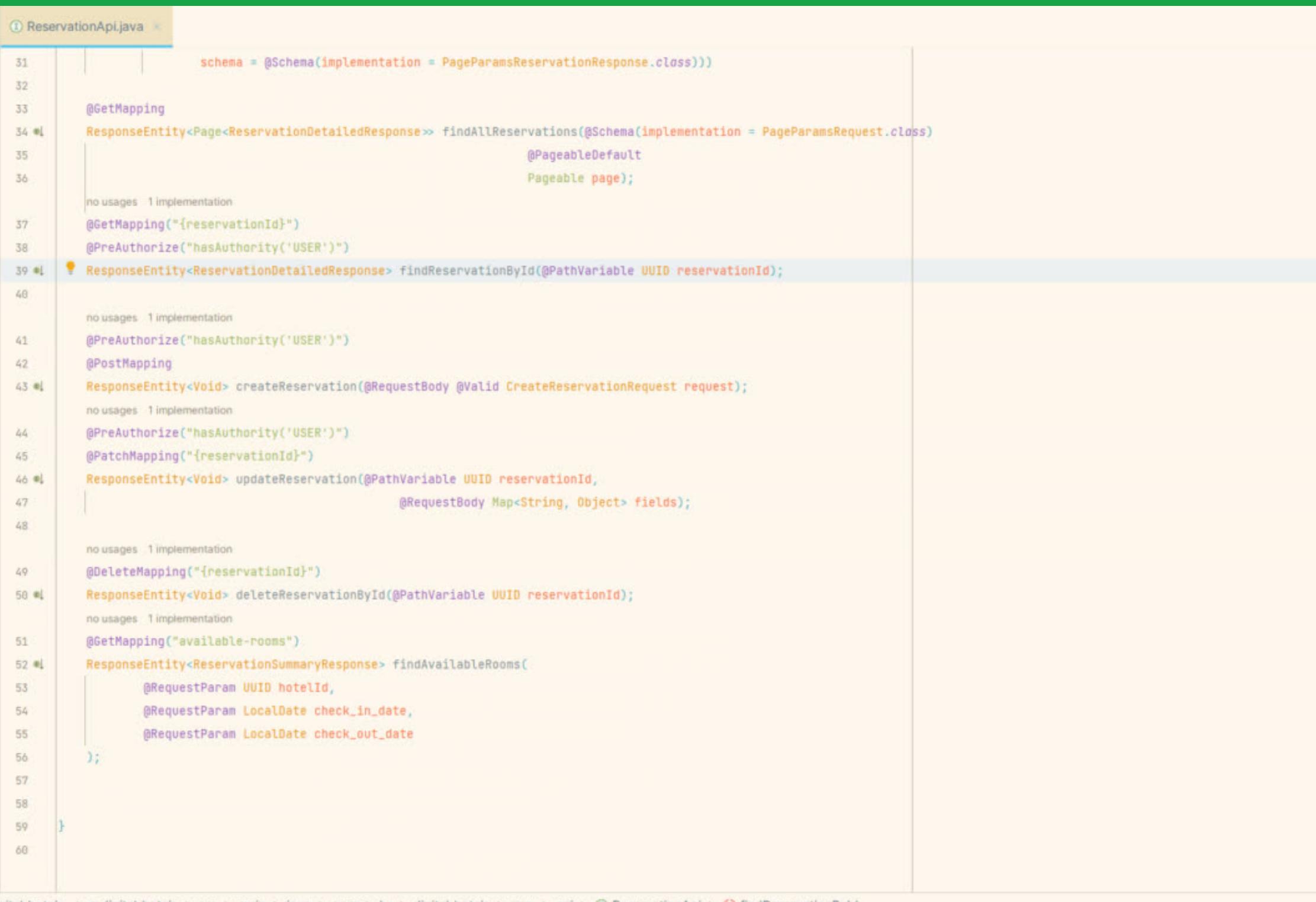
228

229 no usages
230 @Override
231 public ResponseEntity<Page<HostSummaryResponse>> findHostsByHotelId(UUID hotel_id, Pageable pageable) {
232     Page<Host> hostsPage = hostService.findHostsByHotelId(hotel_id, pageable);
233     Page<HostSummaryResponse> response = hostsPage.map( host →
234         new HostSummaryResponse(host.getId(), host.getName(), host.getSurname());
235     return ResponseEntity.ok(response);
236 }
237

238 no usages
239 @Override
240 public ResponseEntity<Page<GuestSummaryResponse>> findGuestByHotelId(UUID hotel_id, Pageable pageable) {
241     Page<Guest> guestsPage = guestService.findGuestsByHotelId(hotel_id, pageable);
242     Page<GuestSummaryResponse> response = guestsPage.map(guest →
243         new GuestSummaryResponse(
244             guest.getId(),
245             guest.getName(),
246             guest.getSurname()
247         );
248     return ResponseEntity.ok(response);
249 }
```

BACK-END

DETALHES DO QUE FEITO

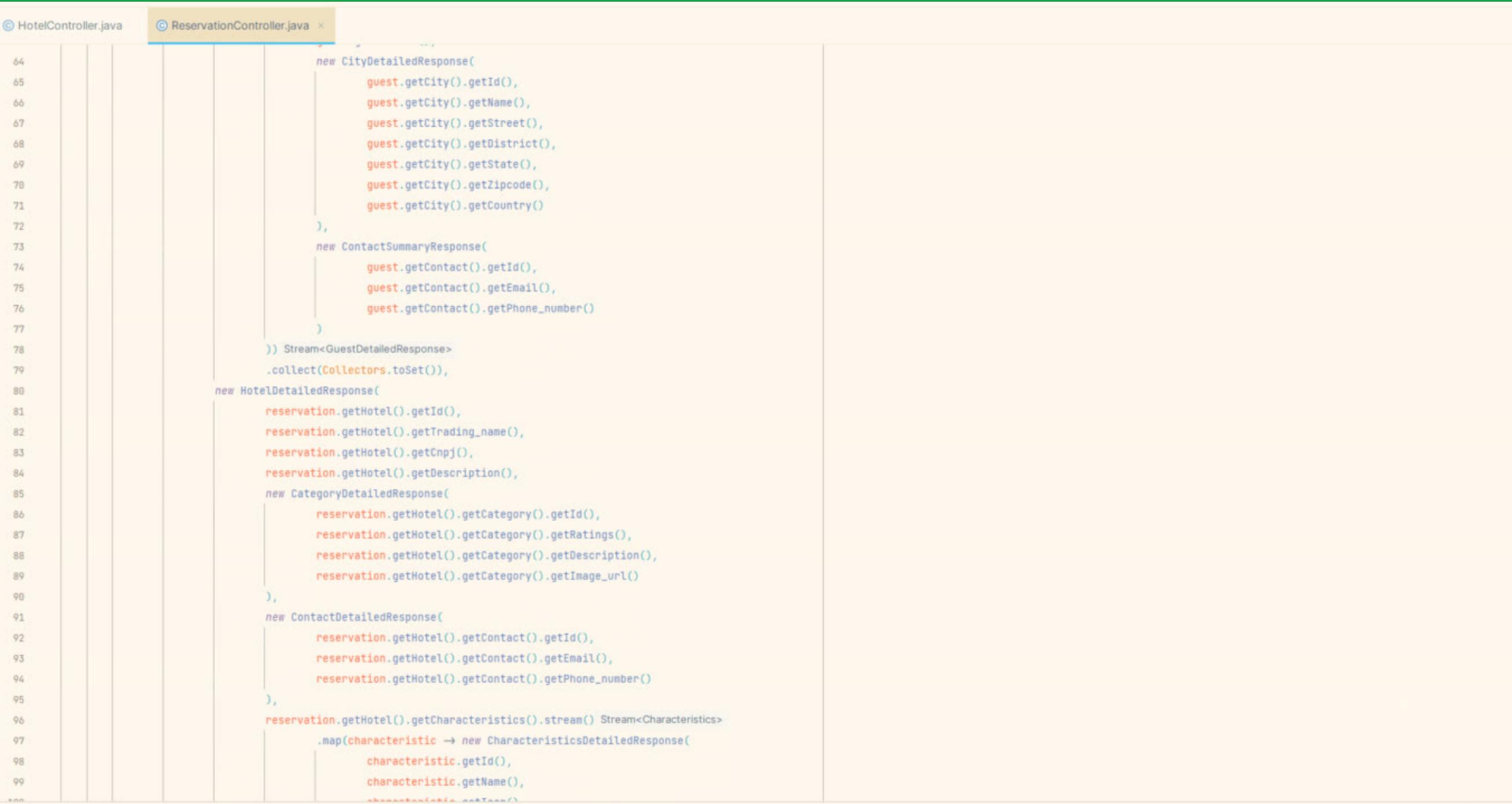


```
① ReservationApi.java ×
31     schema = @Schema(implementation = PageParamsReservationResponse.class))
32
33     @GetMapping
34     @ResponseEntity<Page<ReservationDetailedResponse>> findAllReservations(@Schema(implementation = PageParamsRequest.class)
35                                         @PageableDefault
36                                         Pageable page);
37
38     no usages 1 implementation
39     @GetMapping("/{reservationId}")
40     @PreAuthorize("hasAuthority('USER')")
41     @ResponseEntity<ReservationDetailedResponse> findReservationById(@PathVariable UUID reservationId);
42
43     no usages 1 implementation
44     @PreAuthorize("hasAuthority('USER')")
45     @PostMapping
46     @ResponseEntity<Void> createReservation(@RequestBody @Valid CreateReservationRequest request);
47
48     no usages 1 implementation
49     @PreAuthorize("hasAuthority('USER')")
50     @PatchMapping("{reservationId}")
51     @ResponseEntity<Void> updateReservation(@PathVariable UUID reservationId,
52                                             @RequestBody Map<String, Object> fields);
53
54     no usages 1 implementation
55     @DeleteMapping("{reservationId}")
56     @ResponseEntity<Void> deleteReservationById(@PathVariable UUID reservationId);
57
58     no usages 1 implementation
59     @GetMapping("available-rooms")
60     @ResponseEntity<ReservationSummaryResponse> findAvailableRooms(
61         @RequestParam UUID hotelId,
62         @RequestParam LocalDate check_in_date,
63         @RequestParam LocalDate check_out_date
64     );
65
66 }
```

Digital Hotel API > digital-hotel-api > src > main > java > com > br > digital-hotel-api > app > api > [ReservationApi.java](#) > [findReservationById](#)

BACK-END

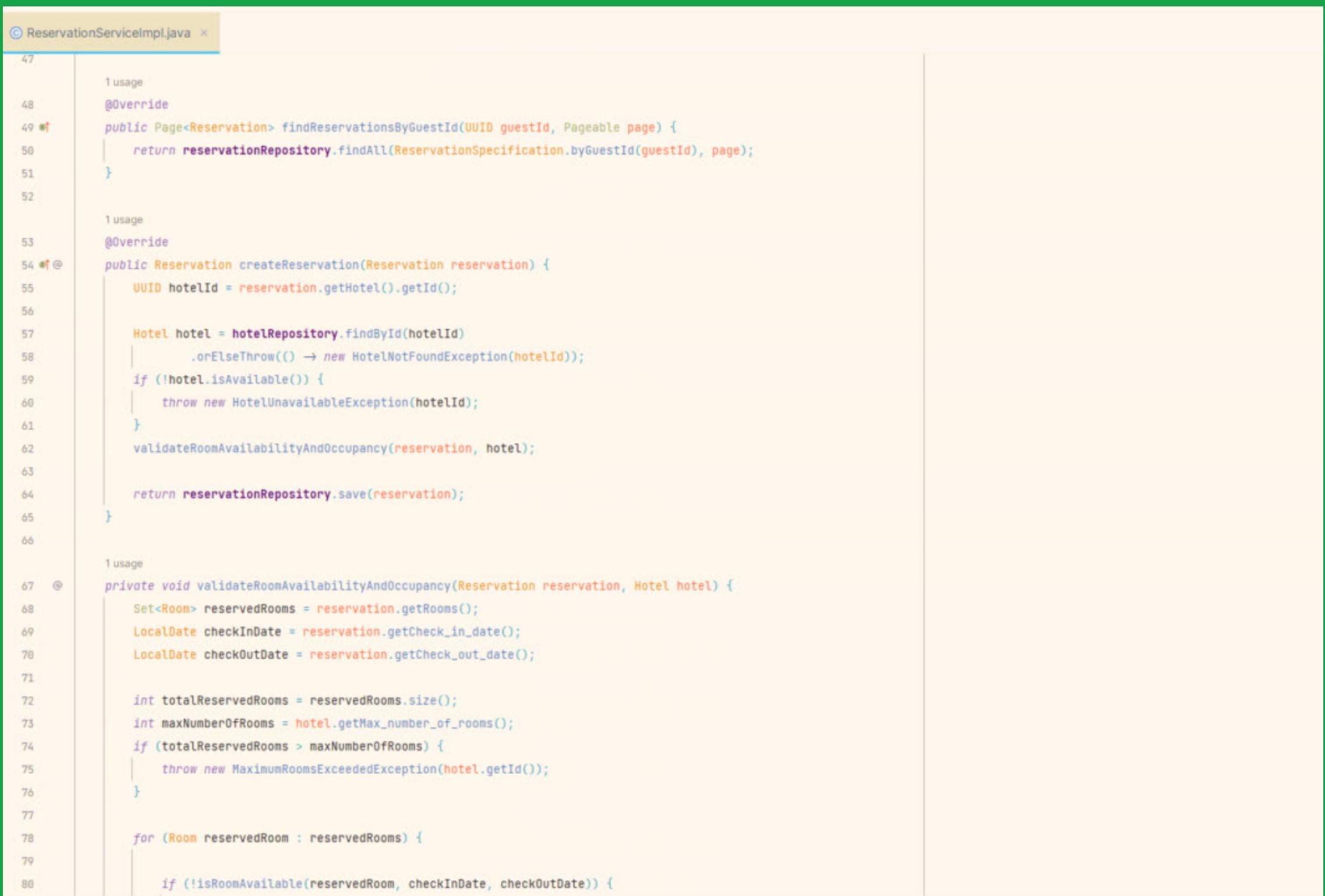
DETALHES DO QUE FEITO



```
© HotelController.java © ReservationController.java x
  ...
  new CityDetailedResponse(
    guest.getCity().getId(),
    guest.getCity().getName(),
    guest.getCity().getStreet(),
    guest.getCity().getDistrict(),
    guest.getCity().getState(),
    guest.getCity().getZipcode(),
    guest.getCity().getCountry()
  ),
  new ContactSummaryResponse(
    guest.getContact().getId(),
    guest.getContact().getEmail(),
    guest.getContact().getPhone_number()
)
)) Stream<GuestDetailedResponse>
.collect(Collectors.toSet()),
new HotelDetailedResponse(
  reservation.getHotel().getId(),
  reservation.getHotel().getTrading_name(),
  reservation.getHotel().getCnpj(),
  reservation.getHotel().getDescription(),
  new CategoryDetailedResponse(
    reservation.getHotel().getCategory().getId(),
    reservation.getHotel().getCategory().getRatings(),
    reservation.getHotel().getCategory().getDescription(),
    reservation.getHotel().getCategory().getImage_url()
  ),
  new ContactDetailedResponse(
    reservation.getHotel().getContact().getId(),
    reservation.getHotel().getContact().getEmail(),
    reservation.getHotel().getContact().getPhone_number()
),
reservation.getHotel().getCharacteristics().stream() Stream<Characteristics>
.map(characteristic -> new CharacteristicsDetailedResponse(
  characteristic.getId(),
  characteristic.getName(),
  characteristic.getDescription(),
  characteristic.getValue()
))
)
```

BACK-END

DETALHES DO QUE FEITO



```
ReservationServiceImpl.java
47     1 usage
48
49     @Override
50     public Page<Reservation> findReservationsByGuestId(UUID guestId, Pageable page) {
51         return reservationRepository.findAll(ReservationSpecification.byGuestId(guestId), page);
52     }
53
54     1 usage
55     @Override
56     @Transactional
57     public Reservation createReservation(Reservation reservation) {
58         UUID hotelId = reservation.getHotel().getId();
59
60         Hotel hotel = hotelRepository.findById(hotelId)
61             .orElseThrow(() -> new HotelNotFoundException(hotelId));
62         if (!hotel.isAvailable()) {
63             throw new HotelUnavailableException(hotelId);
64         }
65         validateRoomAvailabilityAndOccupancy(reservation, hotel);
66
67         return reservationRepository.save(reservation);
68     }
69
70     1 usage
71     @Transactional
72     private void validateRoomAvailabilityAndOccupancy(Reservation reservation, Hotel hotel) {
73         Set<Room> reservedRooms = reservation.getRooms();
74         LocalDate checkInDate = reservation.getCheck_in_date();
75         LocalDate checkOutDate = reservation.getCheck_out_date();
76
77         int totalReservedRooms = reservedRooms.size();
78         int maxNumberOfRooms = hotel.getMax_number_of_rooms();
79         if (totalReservedRooms > maxNumberOfRooms) {
80             throw new MaximumRoomsExceededException(hotel.getId());
81         }
82
83         for (Room reservedRoom : reservedRooms) {
84             if (!isRoomAvailable(reservedRoom, checkInDate, checkOutDate)) {
```

BACK-END

DETALHES DO QUE FEITO

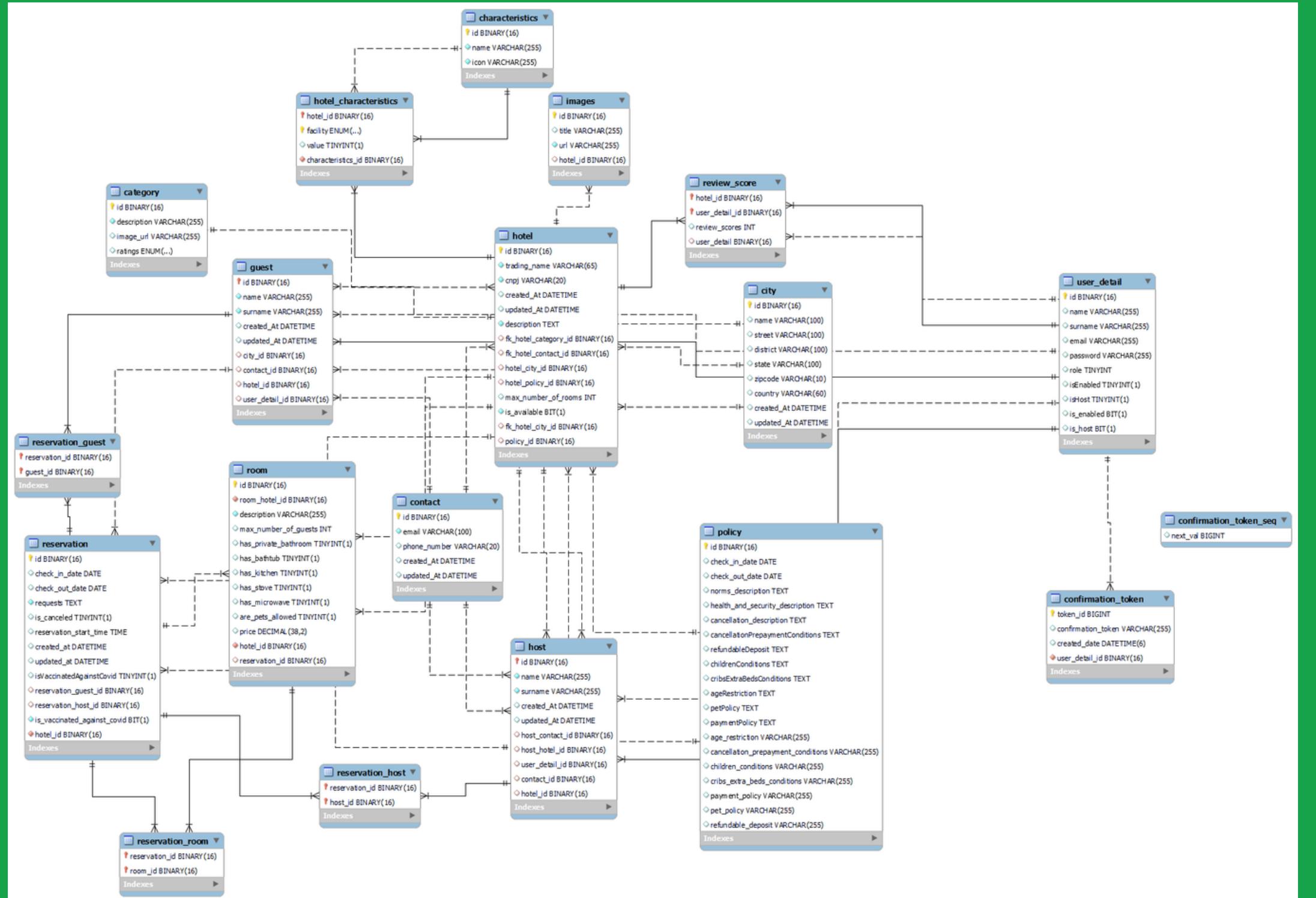
```
ReservationServiceImpl.java
47
48     1 usage
49     @Override
50     public Page<Reservation> findReservationsByGuestId(UUID guestId, Pageable page) {
51         return reservationRepository.findAll(ReservationSpecification.byGuestId(guestId), page);
52     }
53
54     1 usage
55     @Override
56     public Reservation createReservation(Reservation reservation) {
57         UUID hotelId = reservation.getHotel().getId();
58
59         Hotel hotel = hotelRepository.findById(hotelId)
60             .orElseThrow(() -> new HotelNotFoundException(hotelId));
61         if (!hotel.isAvailable()) {
62             throw new HotelUnavailableException(hotelId);
63         }
64         validateRoomAvailabilityAndOccupancy(reservation, hotel);
65
66         return reservationRepository.save(reservation);
67     }
68
69     1 usage
70     private void validateRoomAvailabilityAndOccupancy(Reservation reservation, Hotel hotel) {
71         Set<Room> reservedRooms = reservation.getRooms();
72         LocalDate checkInDate = reservation.getCheck_in_date();
73         LocalDate checkOutDate = reservation.getCheck_out_date();
74
75         int totalReservedRooms = reservedRooms.size();
76         int maxNumberOfRooms = hotel.getMax_number_of_rooms();
77         if (totalReservedRooms > maxNumberOfRooms) {
78             throw new MaximumRoomsExceededException(hotel.getId());
79         }
80
81         for (Room reservedRoom : reservedRooms) {
82             if (!isRoomAvailable(reservedRoom, checkInDate, checkOutDate)) {
83                 throw new RoomNotAvailableException(reservedRoom.getId());
84             }
85         }
86     }
87
88     1 usage
89     private boolean isRoomAvailable(Room room, LocalDate checkInDate, LocalDate checkOutDate) {
90         // Implementation logic here
91     }
92 }
```

Foram criados métodos para:

- achar o Hotel pela id da Cidade;
- listar todos os Hotéis;
- achar o Hotel pela sua id;
- criação de Hotéis;
- para atualização um Hotel existente;
- deletar um Hotel por usa ID;
- achar os Hotéis pela id da Categoria;
- para um usuário adicionar uma pontuação/classificação a um Hotel por id;
- para adicionar características a um hotel;
- para adicionar Imagens a um determinado Hotel por ID;
- para adicionar Políticas a um Hotel por ID;
- para achar um Hotel através de determinada Política;

BACK-END

DETALHES DO QUE FEITO - DIAGRAMA UML



BACK-END

DETALHES DO QUE FEITO - DIAGRAMA UML

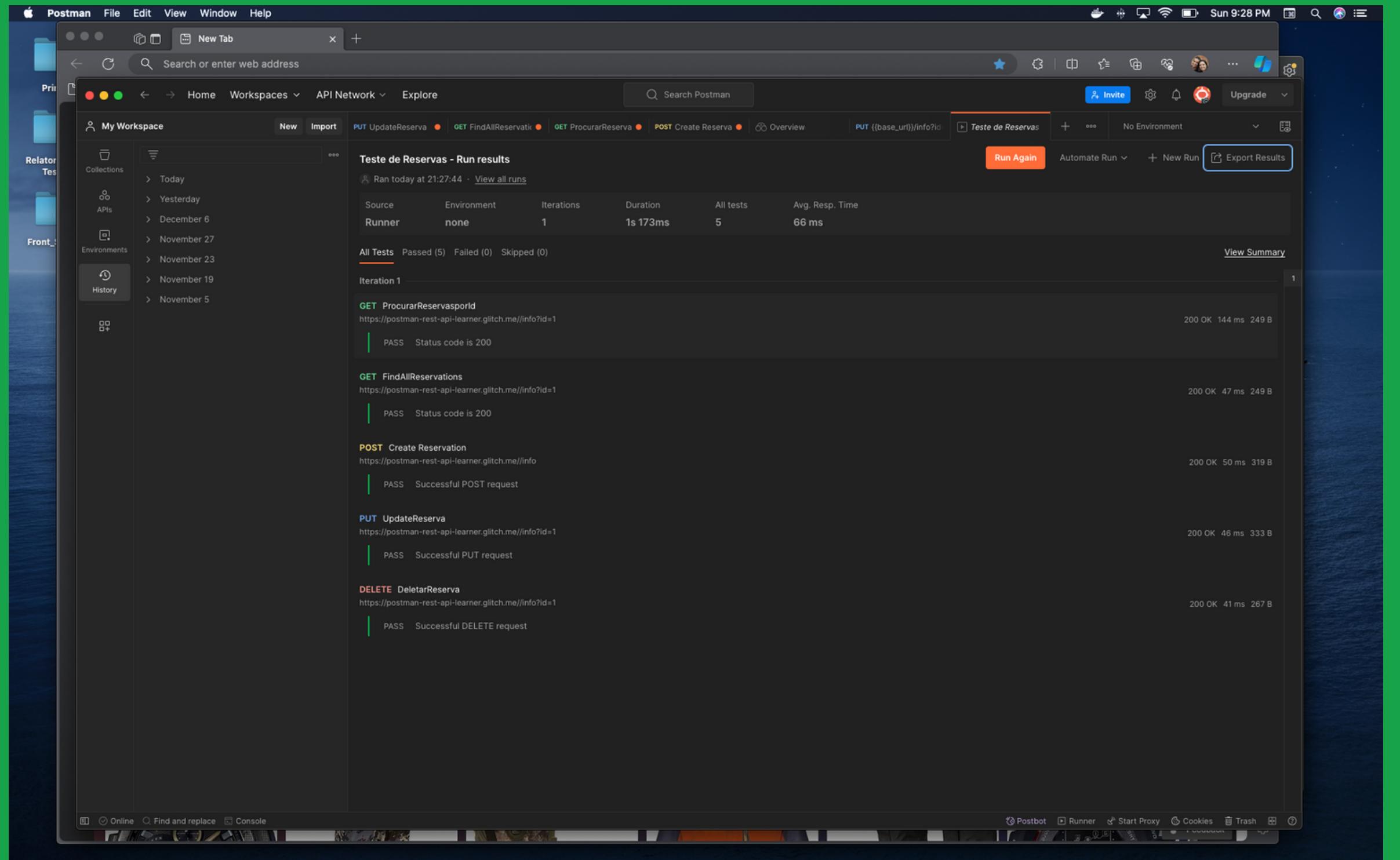


The screenshot shows a code editor window with a tab labeled "create.sql". The content of the file is a series of SQL commands for creating tables and defining foreign keys. The code is numbered from 144 to 188.

```
144     created_At DATETIME,
145     updated_At DATETIME,
146     host_contact_id BINARY(16),
147     host_hotel_id BINARY(16),
148     user_detail_id BINARY(16),
149     FOREIGN KEY (host_contact_id) REFERENCES contact(id),
150     FOREIGN KEY (host_hotel_id) REFERENCES hotel(id),
151     FOREIGN KEY (user_detail_id) REFERENCES user_detail(id)
152 );
153
154
155
156 CREATE TABLE images (
157     id BINARY(16) PRIMARY KEY,
158     title VARCHAR(255),
159     url VARCHAR(255) NOT NULL,
160     hotel_id BINARY(16),
161     FOREIGN KEY (hotel_id) REFERENCES hotel(id)
162 );
163
164
165 CREATE TABLE reservation (
166     id BINARY(16) PRIMARY KEY,
167     check_in_date DATE,
168     check_out_date DATE,
169     requests TEXT NOT NULL,
170     is_canceled BOOLEAN DEFAULT false,
171     reservation_start_time TIME,
172     created_at DATETIME,
173     updated_at DATETIME,
174     isVaccinatedAgainstCovid BOOLEAN DEFAULT false,
175     reservation_guest_id BINARY(16),
176     reservation_host_id BINARY(16),
177     FOREIGN KEY (reservation_guest_id) REFERENCES guest(id),
178     FOREIGN KEY (reservation_host_id) REFERENCES host(id),
179     INDEX (reservation_guest_id),
180     INDEX (reservation_host_id)
```

TESTES

DETALHES DO QUE FEITO



Testes foram escritos e executados na ferramenta Postman, fazendo uma junção das informações constantes na IDE, e no banco de dados, e embora meu conhecimento da ferramenta fosse baixo, contei com o Auxílio da Patrícia Delvequi na formatação das ferramentas, o que possibilitou a execução devida dos testes.

TESTES

DETALHES DO QUE FEITO

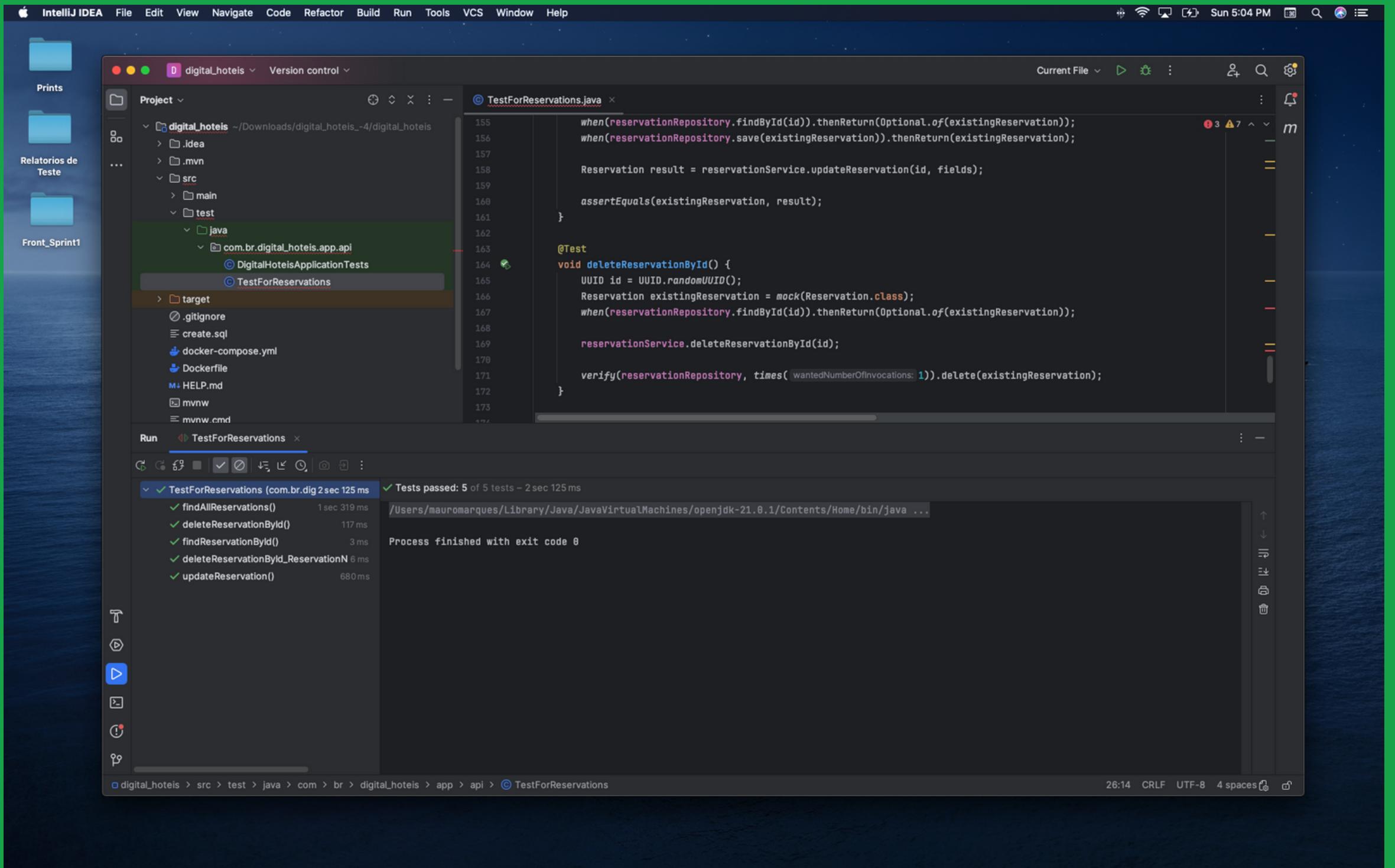
The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure under "grupo-5-Testing".
- Code Editor:** The file `LoginServiceTest.java` is open, containing test cases for the `LoginService` class.
- Run Tab:** The "LoginServiceTest" run configuration is selected.
- Output Window:** Displays the test results:
 - Tests passed: 3 of 3 tests - 54 ms
 - testFailedLogin() (48 ms)
 - testSuccessfulLogin() (3 ms)
 - testInvalidCredentials() (3 ms)
- Status Bar:** Shows the current file as `LoginServiceTest.java`, the date as `Thu 7:57 PM`, and the exit code as `0`.

Na parte de Front-End do projeto, foram realizadas testes exploratórios nas funcionalidades de Login redimensionamento, ciclagem dos botões, alternância dos modos claro e escuro e visualização correta dos itens de estilo e imagem.

TESTES

DETALHES DO QUE FEITO



The screenshot shows the IntelliJ IDEA interface with a Java test file open and a successful run output.

Project Structure: The left sidebar shows the project structure for "digital_hoteis". It includes a "src" folder with "main" and "test" subfolders. The "test/java/com.br.digital_hoteis.app.api" package contains two classes: "DigitalHotelsApplicationTests" and "TestForReservations".

Code Editor: The main editor window displays the "TestForReservations.java" file. The code implements tests for a reservation service using Mockito and JUnit. It includes methods to update, delete, and verify reservations.

```
when(reservationRepository.findById(id)).thenReturn(Optional.of(existingReservation));
when(reservationRepository.save(existingReservation)).thenReturn(existingReservation);

Reservation result = reservationService.updateReservation(id, fields);

assertEquals(existingReservation, result);
}

@Test
void deleteReservationById() {
UUID id = UUID.randomUUID();
Reservation existingReservation = mock(Reservation.class);
when(reservationRepository.findById(id)).thenReturn(Optional.of(existingReservation));

reservationService.deleteReservationById(id);

verify(reservationRepository, times(wantedNumberOfInvocations)).delete(existingReservation);
}
```

Run Results: The bottom panel shows the run results for "TestForReservations". It lists five test methods: "findAllReservations()", "deleteReservationById()", "findReservationById()", "deleteReservationByid_ReservatioN()", and "updateReservation()". All tests passed in 2 seconds.

Test Method	Time
findAllReservations()	1 sec 319 ms
deleteReservationById()	117 ms
findReservationById()	3 ms
deleteReservationByid_ReservatioN()	6 ms
updateReservation()	680 ms

The status bar at the bottom indicates the run was completed with exit code 0.

TESTES

DETALHES DO QUE FEITO

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project Structure:** The project is named "Testes [grupo-5]". It contains a "backend" module with a "digital_hoteis" package. The "src/test/java/com.br.digital_hoteis.app.api" directory contains the test class "DigitalHoteisApplicationTests.java".
- Code Editor:** The file "DigitalHoteisApplicationTests.java" is open, showing test methods like `deleteCategoryById()`, `findAllCategories()`, `findCategoryById()`, `deleteCategoryById_CategoryNotFound()`, `updateCategory()`, and `createCategory()`. The code uses Mockito annotations such as `@InjectMocks`, `@Mock`, and `MockitoAnnotations.openMocks(testClass);`.
- Run Tab:** The "Run" tab shows the test class "DigitalHoteisApplicationTests" and its methods. The method `findCategoryById()` is currently selected.
- Toolbars and Status Bar:** The status bar at the bottom shows "41:31 LF UTF-8 4 spaces". A message from ChatGPT is visible at the bottom center: "ChatGPT can make mistakes. Consider checking important information."

TESTES

DETALHES DO QUE FEITO

The screenshot shows the IntelliJ IDEA interface with the following details:

- Project View:** Shows the project structure for "digital_hoteis_". The "specification" package contains several classes: `NotFoundException`, `ReservationNotFoundException`, `RoomNotFoundException`, `AddressRepository`, `CategoryRepository`, `ContactRepository`, `GuestRepository`, `HostRepository`, `HotelRepository`, `ReservationRepository`, `RoomRepository`, and `UserRepository`. A specific class, `UserDetailSpecification`, is selected.
- Code Editor:** Displays the `JwtService.java` file. The code defines a public interface `JwtService` with three methods: `extractUserName(String token)`, `generateToken(UserDetails userDetails)`, and `boolean isTokenValid(String token, UserDetails userDetails)`.
- Run Tab:** Shows the results of the `DigitalHoteisApplicationTests` run. It indicates 6 tests passed in 3 seconds and 302 milliseconds. The tests listed are: `deleteCategoryById()`, `findAllCategories()`, `findCategoryById()`, `deleteCategoryById_CategoryNotFound()`, `updateCategory()`, and `createCategory()`.
- Output Tab:** Displays the command-line output of the test run. It includes Java agent loading warnings, OpenJDK 64-Bit Server VM warnings, and a final message: "Process finished with exit code 0".
- Status Bar:** Shows the current file as `DigitalHoteisApplicationTests.java`, the time as 5:18, and encoding as UTF-8.

FRONT-END

DETALHES DO QUE FEITO

Iniciar sessão

E-mail

Senha

Entrar

Ainda não tem conta [Registre-se](#)

Hotéis

Conrad Maldives Rangali Island

★★★★ Ilha Rangali, Maldivas



Localizado em 2 ilhas separadas, conectadas por uma ponte, o luxuoso Conrad Maldives Rangali Island dispõe de villas privativas e espaçosas. A acomodação tem restaurante subaquático todo de vidro, o Ithaai Restaurant, e 2 spas premiados.



-  Mergulho com snorkel
-  Instalações para esportes aquáticos no local
-  Pesca
-  Wi-fi gratuito
-  Café da Manhã

Regras da Casa

Check-in/Check-out
Check-in a qualquer momento após as 15:00 e check-out a qualquer momento antes das 12:00.

Login e Register

Administração

Detalhes dos Produtos

FRONT-END

DETALHES DO QUE FEITO

The screenshot shows a web browser window for 'localhost:5173' displaying the 'Five Hotels' website. The page features a navigation bar with a logo, 'Login', and 'Cadastre-se' buttons. A search bar is present with the placeholder 'Aracaju'. Below the search bar is a section titled 'Buscar ofertas em hotéis, casa e muito mais' with a 'Buscar' button. A 'Busque por categorias' section displays four categories: 'Apartamentos' (with an image of a modern interior), 'Pousadas' (with an image of colorful wooden bungalows), 'Hostels' (with an image of a hallway), and 'Hotéis' (with an image of a modern hotel building). At the bottom, there's a 'Recomendações' section showing two cards for 'Hotel Atalaia' and one for 'Hotel Arvoredo', each with an image, name, location, and a brief description.

Five Hotels

Buscar ofertas em hotéis, casa e muito mais

Aracaju

Buscar

Busque por categorias

Apartamentos

Pousadas

Hostels

Hotéis

Recomendações

Hotel Atalaia

Hotéis

Conforto com vista para o mar. Café da manhã saboroso e bem variado. Quartos com TV, Wifi e ar condicionado. A 100 metros do mar.

Aracaju

Hotel Atalaia

Hotéis

Conforto com vista para o mar. Café da manhã saboroso e bem variado. Quartos com TV, Wifi e ar condicionado. A 100 metros do mar.

Aracaju

Hotel Arvoredo

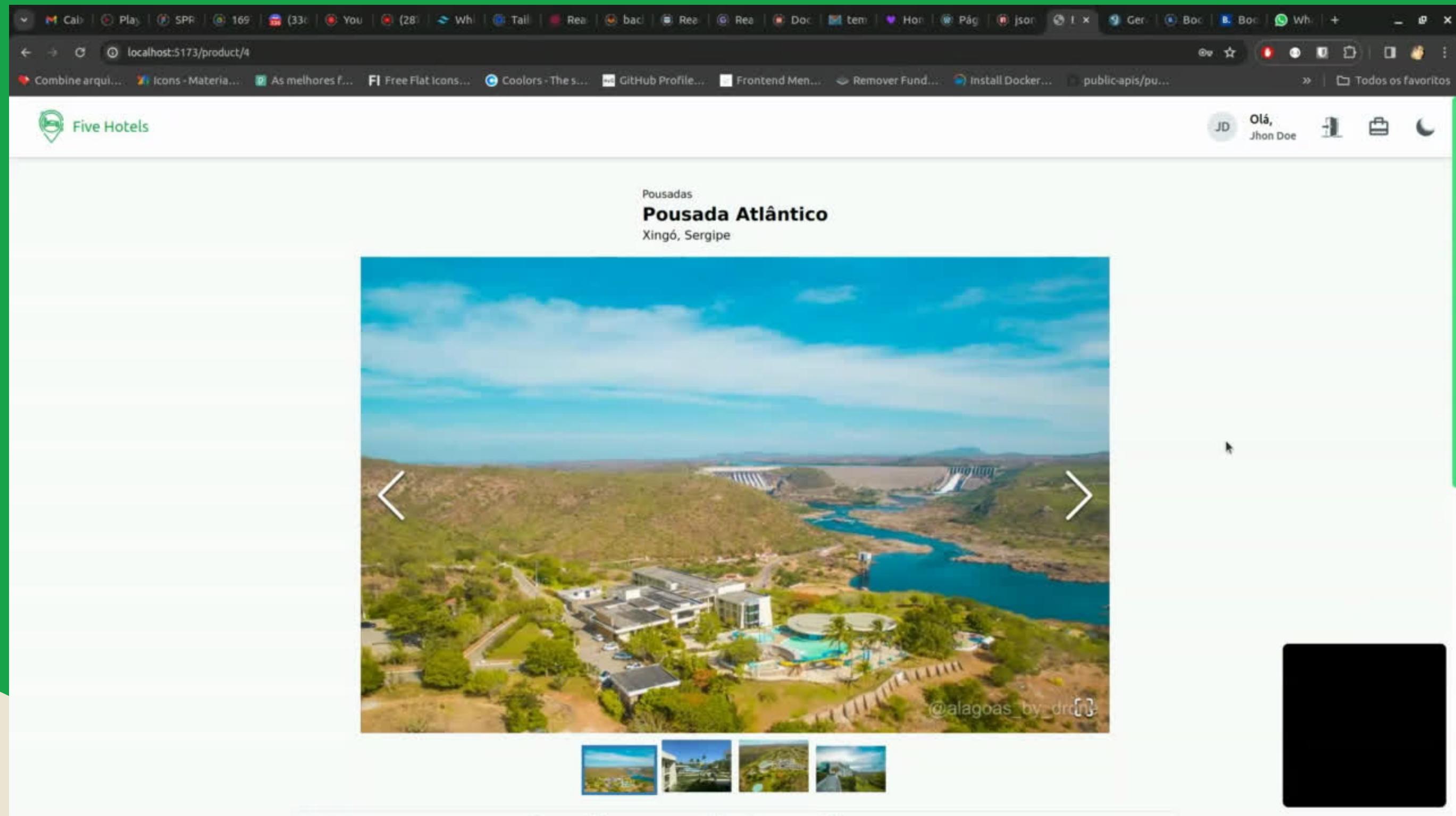
Hotéis

Venha ficar proximo da natureza. Lugar muito aconchegante.

Aracaju

FRONT-END

DETALHES DO QUE FEITO



FRONT-END

DETALHES DO QUE FEITO

