FIVE HOTELS

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

December 2023

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

Five Hotels

BACK-END

DETALHES DO QUE FEITO

BACK-END

DETALHES DO QUE FEITO

```
spring.mvc.cors.allowed-origins=http://localhost:5173

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.datasource.url=jdbc:mysql://mysql:3306/digital_hoteis?useTimezone=true&serverTimezone=America/Sao_Paulo
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
springdoc.swagger-ui.path=/swagger-ui.html
spring.main.allow-bean-definition-overriding=true
spring.jpa.properties.hibernate.format_sql=true

# Confirmation email's configuration
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=sacdigitalhoteis@gmail.com
spring.mail.password=bxqzqhjisfhxaydd
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true

# SQL statements
logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

logging.level.root=debug
```

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

AuthenticationController.java

```
            log.error("Error during sign-up: Bad request - {}", e.getMessage(), e);
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
        } catch (HttpClientErrorException e) {
            if (e.getStatusCode() == HttpStatus.CONFLICT) {
                log.error("Error during sign-up: Conflict - Email already in use. Email: {}", request.email(), e);
                return ResponseEntity.status(HttpStatus.CONFLICT)
                        .body("Infelizmente não foi possivel registrar. Por favor, tente novamente mais tarde");
            } else {
                log.error("Error during sign-up: HTTP Status - {}. Email: {}", e.getStatusCode(), request.email(), e);
                return ResponseEntity.status(e.getStatusCode()).body("Ocorreu um erro: " + e.getMessage());
            }
        } catch (Exception e) {
            log.error("Error during sign-up: Internal Server Error. Email: {}", request.email(), e);
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Erro interno do servidor: " + e.getMessage());
        }
    }

    no usages
    @RequestMapping(value = "/confirm-account", method = {RequestMethod.GET, RequestMethod.POST})
    public ResponseEntity<?> confirmUserAccount(@RequestParam("token") String confirmationToken) {
        ConfirmationToken token = confirmationTokenRepository.findByConfirmationToken(confirmationToken);

        if (token != null) {
            UserDetail user = token.getUserEntity();

            if (!user.isEnabled()) {
                user.setEnabled(true);
                String jwt = String.valueOf(authenticationService.signIn(new SignIn(user.getEmail(),  password: null), String.valueOf(token)));


                return ResponseEntity.ok( body: "Email successfully verified!");
            } else {
                return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("User is already confirmed.");
            }
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid or expired token.");
        }
```

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

BACK-END

DETALHES DO QUE FEITO

AuthenticationServiceImpl.java

```java
public String signUp(SignUp request) {
    if (!request.getEmail().equals(request.getRetype_email())) {
        throw new IllegalArgumentException("Error: The email and retype_email must match.");
    }
    UserDetail user = UserDetail.builder()
            .name(request.getName())
            .surname(request.getSurname())
            .email(request.getEmail())
            .password(passwordEncoder.encode(request.getPassword()))
            .isHost(request.isHost())
            .build();
    if (userRepository.existsByEmail(user.getEmail())) {
        throw new IllegalArgumentException("Erro: O e-mail já está em uso");
    }
    if (user.isHost()) {
        userRepository.saveHost(user.getEmail(), UserPermissionEnum.ADMIN, user.getName(), user.getSurname());
    } else {
        userRepository.saveGuest(user.getEmail(), UserPermissionEnum.USER, user.getName(), user.getSurname());
    }
    ConfirmationToken confirmationToken = new ConfirmationToken(user);
    confirmationTokenRepository.save(confirmationToken);

    SimpleMailMessage mailMessage = new SimpleMailMessage();
    mailMessage.setTo(user.getEmail());
    mailMessage.setSubject("Conclua seu cadastro");
    mailMessage.setText("Para confirmar sua conta, clique no link por favor: "
            + "<a href='http://localhost:9090/v1/authentication/confirm-account?token="
            + confirmationToken.getConfirmationToken() + "'>Clique aqui</a>");
    emailService.sendEmail(mailMessage);

    return confirmationToken.getConfirmationToken();
}

no usages
@Override
public ResponseEntity<?> confirmUserAccount(String confirmationToken) {
```

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

BACK-END

DETALHES DO QUE FEITO

AuthenticationServiceImpl.java

```java
public String signUp(SignUp request) {
    if (!request.getEmail().equals(request.getRetype_email())) {
        throw new IllegalArgumentException("Error: The email and retype_email must match.");
    }
    UserDetail user = UserDetail.builder()
            .name(request.getName())
            .surname(request.getSurname())
            .email(request.getEmail())
            .password(passwordEncoder.encode(request.getPassword()))
            .isHost(request.isHost())
            .build();
    if (userRepository.existsByEmail(user.getEmail())) {
        throw new IllegalArgumentException("Erro: O e-mail já está em uso");
    }
    if (user.isHost()) {
        userRepository.saveHost(user.getEmail(), UserPermissionEnum.ADMIN, user.getName(), user.getSurname());
    } else {
        userRepository.saveGuest(user.getEmail(), UserPermissionEnum.USER, user.getName(), user.getSurname());
    }
    ConfirmationToken confirmationToken = new ConfirmationToken(user);
    confirmationTokenRepository.save(confirmationToken);

    SimpleMailMessage mailMessage = new SimpleMailMessage();
    mailMessage.setTo(user.getEmail());
    mailMessage.setSubject("Conclua seu cadastro");
    mailMessage.setText("Para confirmar sua conta, clique no link por favor: "
            + "<a href='http://localhost:9090/v1/authentication/confirm-account?token="
            + confirmationToken.getConfirmationToken() + "'>Clique aqui</a>");
    emailService.sendEmail(mailMessage);

    return confirmationToken.getConfirmationToken();
}

no usages
@Override
public ResponseEntity<?> confirmUserAccount(String confirmationToken) {
```

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

BACK-END

DETALHES DO QUE FEITO

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

BACK-END

DETALHES DO QUE FEITO

EmailServiceImpl.java

```java
package com.br.digital_hoteis.domain.service.impl;


import ...


1 usage
@Service("emailService")
public class EmailServiceImpl implements EmailService {
    2 usages
    @Autowired
    private final JavaMailSender javaMailSender;

    no usages
    @Autowired
    public EmailServiceImpl(JavaMailSender javaMailSender) { this.javaMailSender = javaMailSender; }

    2 usages
    @Async
    public void sendEmail(SimpleMailMessage email) { javaMailSender.send(email); }

}
```

JavaMailSender:
Classe que auxilia no envio do email

BACK-END

DETALHES DO QUE FEITO

WebServerSecurityConfiguration.java

```java
    private final UserService userService;

    4 usages
    private static final String[] AUTH_WHITELIST = {
            "/api/v1/auth/**",
            "/v3/api-docs/**",
            "/v3/api-docs.yaml",
            "/swagger-ui/**",
            "/swagger-ui.html",
            "/Categories/**",
            "/v1/authentication/confirm-account",
            "/v1/authentication/**"
    };

    no usages
    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins("http://localhost:5173")
                .allowedMethods("GET", "POST", "PUT", "DELETE")
                .allowedHeaders("*")
                .allowCredentials(true)
                .maxAge(3600);
    }

    no usages
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        http.csrf(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(request ->
                        request
                                .requestMatchers(HttpMethod.POST,  ...patterns: "/v1/authentication/**").permitAll()
                                .requestMatchers(HttpMethod.POST, AUTH_WHITELIST).permitAll()
                                .requestMatchers(HttpMethod.GET, AUTH_WHITELIST).permitAll()
                                .requestMatchers(HttpMethod.PUT, AUTH_WHITELIST).permitAll()
                                .requestMatchers(HttpMethod.DELETE, AUTH_WHITELIST).permitAll()
                                .anyRequest().authenticated())
```

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

BACK-END

DETALHES DO QUE FEITO

HotelApi.java

```java
@GetMapping("{hotelId}/hosts")
@PreAuthorize("hasAuthority('ADMIN')")
ResponseEntity<Page<HostSummaryResponse>> findHostsByHotelId(@PathVariable UUID hotel_id,
                                                             @PageableDefault Pageable pageable);

no usages  1 implementation
@GetMapping("{hotelId}/guests")
@PreAuthorize("hasAuthority('ADMIN')")
ResponseEntity<Page<GuestSummaryResponse>> findGuestByHotelId(@PathVariable UUID hotel_id,
                                                              @PageableDefault Pageable pageable);

no usages  1 implementation
@GetMapping("{hotelId}/policy")
@PreAuthorize("hasAuthority('USER')")
@ApiResponse(responseCode = "200", description = "Retrieve hotel with policy",
        content = @Content(mediaType = MediaType.APPLICATION_JSON_VALUE,
                schema = @Schema(implementation = HotelDetailedResponse.class)))
ResponseEntity<Map<String, Object>> findHotelWithPolicy(@PathVariable UUID hotel_id);

no usages  1 implementation
@PostMapping("{hotelId}/reviews")
@PreAuthorize("hasAuthority('USER')")
ResponseEntity<Void> addReviewScore(@PathVariable UUID hotel_id,
                                    @RequestParam UUID user_id,
                                    @RequestParam Integer review_scores);

no usages  1 implementation
@PostMapping
@PreAuthorize("hasAuthority('ADMIN')")
ResponseEntity<Void> createHotel(@RequestBody @Valid CreateHotelRequest request);

no usages  1 implementation
@PatchMapping("{hotelId}")
ResponseEntity<Void> updateHotel(@PathVariable UUID hotel_id,
                                 @RequestBody Map<String, Object> fields);
no usages  1 implementation
@PostMapping("{hotelId}/policies")
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
        1 usage
87      @Override
88      public void addReviewScore(UUID hotel_id, UUID user_id, Integer review_scores) {
89          if (review_scores < 1 || review_scores > 5) {
90              throw new IllegalArgumentException("Rating must be between 1 and 5.");
91          }
92
93          Hotel hotel = hotelRepository.findById(hotel_id)
94                  .orElseThrow(() -> new HotelNotFoundException(hotel_id));
95
96          if (hotel.getReview_scores() == null) {
97              hotel.setReview_scores(new HashSet<>());
98          }
99
100         ReviewScore reviewScore = ReviewScore.newReviewScore(hotel_id, user_id, review_scores);
101
102         reviewScoreService.setHotelForReviewScore(hotel, reviewScore);
103
104         hotelRepository.save(hotel);
105     }
106
        1 usage
107     public void addCharacteristics(UUID hotel_id, Characteristics characteristics) {
108         Hotel hotel = hotelRepository.findHotelByIdWithCharacteristics(hotel_id);
109
110         if (hotel != null) {
111             hotel.getCharacteristics().add(characteristics);
112             hotelRepository.save(hotel);
113         } else {
114             hotelRepository.findById(hotel_id)
115                     .orElseThrow(() -> new HotelNotFoundException(hotel_id));
116         }
117     }
118
        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
        no usages
238     @Override
239     public ResponseEntity<Page<GuestSummaryResponse>> findGuestByHotelId(UUID hotel_id, Pageable pageable) {
240         Page<Guest> guestsPage = guestService.findGuestsByHotelId(hotel_id, pageable);
241         Page<GuestSummaryResponse> response = guestsPage.map(guest →
242                 new GuestSummaryResponse(
243                         guest.getId(),
244                         guest.getName(),
245                         guest.getSurname()
246                 ));
247         return ResponseEntity.ok(response);
```

BACK-END

DETALHES DO QUE FEITO

BACK-END

DETALHES DO QUE FEITO

BACK-END

DETALHES DO QUE FEITO

ReservationServiceImpl.java

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

BACK-END

DETALHES DO QUE FEITO

ReservationServiceImpl.java

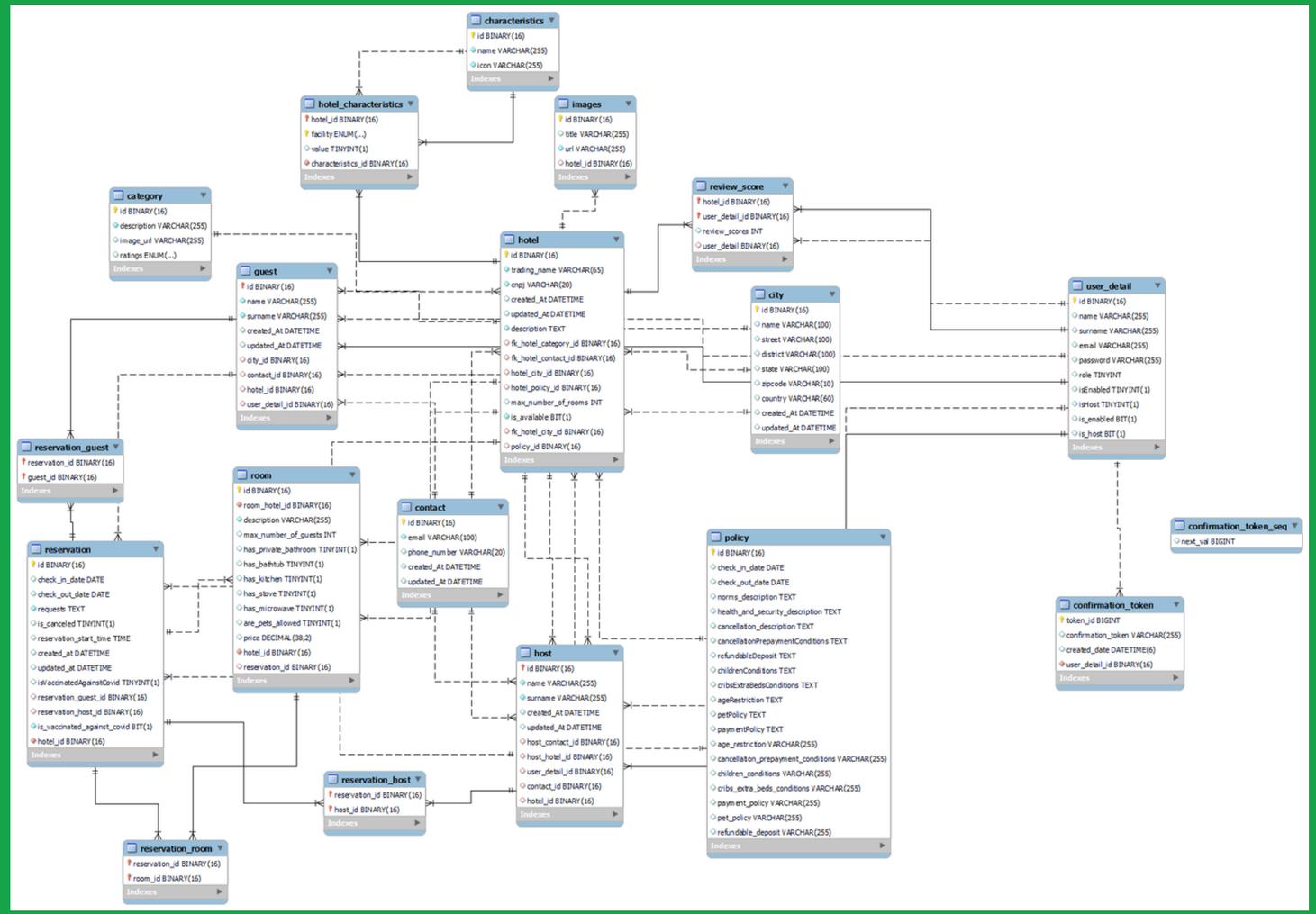
```java
1 usage
@Override
public Page<Reservation> findReservationsByGuestId(UUID guestId, Pageable page) {
    return reservationRepository.findAll(ReservationSpecification.byGuestId(guestId), page);
}

1 usage
@Override
public Reservation createReservation(Reservation reservation) {
    UUID hotelId = reservation.getHotel().getId();

    Hotel hotel = hotelRepository.findById(hotelId)
            .orElseThrow(() -> new HotelNotFoundException(hotelId));
    if (!hotel.isAvailable()) {
        throw new HotelUnavailableException(hotelId);
    }
    validateRoomAvailabilityAndOccupancy(reservation, hotel);

    return reservationRepository.save(reservation);
}

1 usage
private void validateRoomAvailabilityAndOccupancy(Reservation reservation, Hotel hotel) {
    Set<Room> reservedRooms = reservation.getRooms();
    LocalDate checkInDate = reservation.getCheck_in_date();
    LocalDate checkOutDate = reservation.getCheck_out_date();

    int totalReservedRooms = reservedRooms.size();
    int maxNumberOfRooms = hotel.getMax_number_of_rooms();
    if (totalReservedRooms > maxNumberOfRooms) {
        throw new MaximumRoomsExceededException(hotel.getId());
    }

    for (Room reservedRoom : reservedRooms) {

        if (!isRoomAvailable(reservedRoom, checkInDate, checkOutDate)) {
```

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;

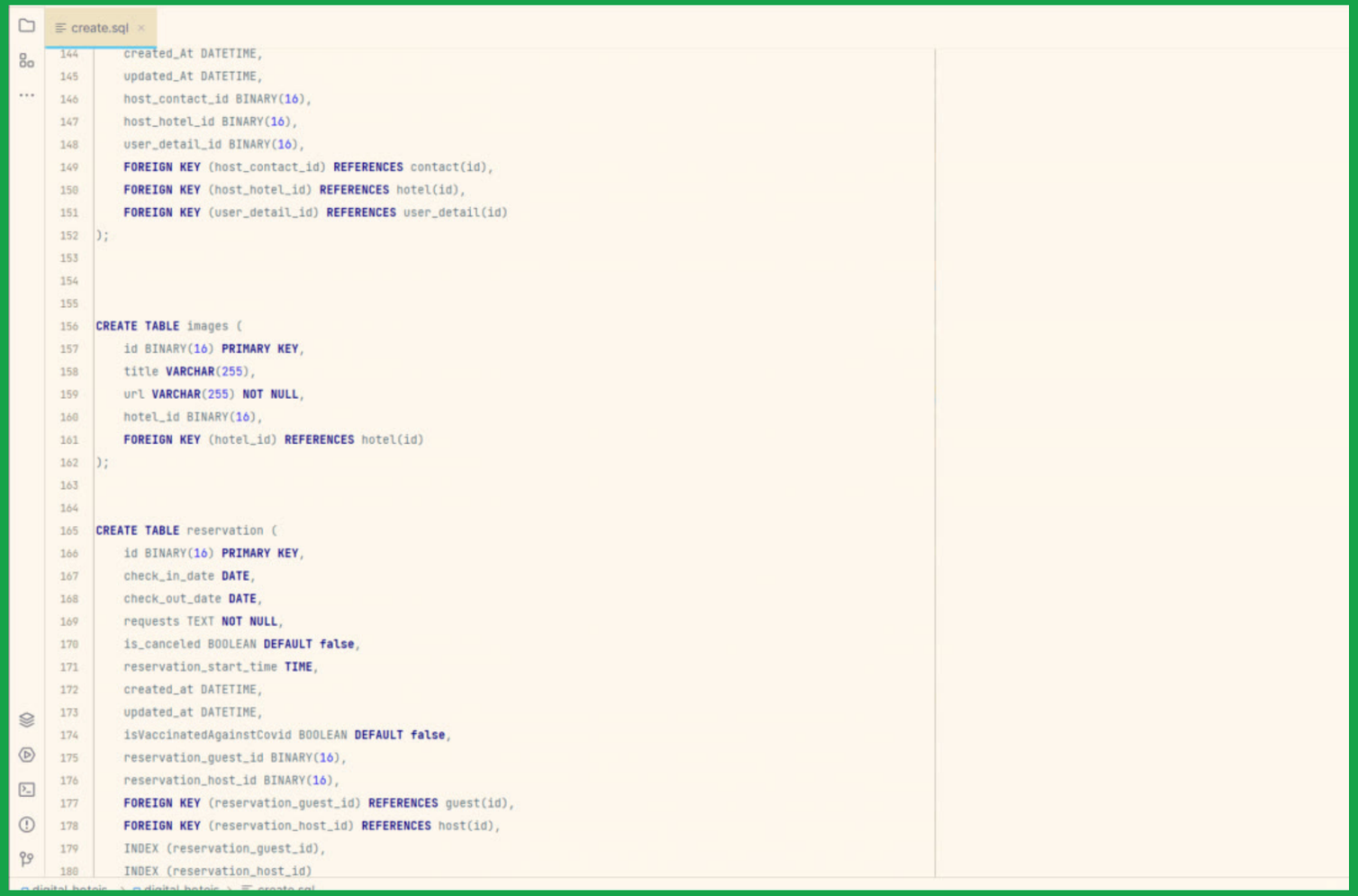
BACK-END

DETALHES DO QUE FEITO - DIAGRAMA UML



BACK-END

DETALHES DO QUE FEITO - DIAGRAMA UML

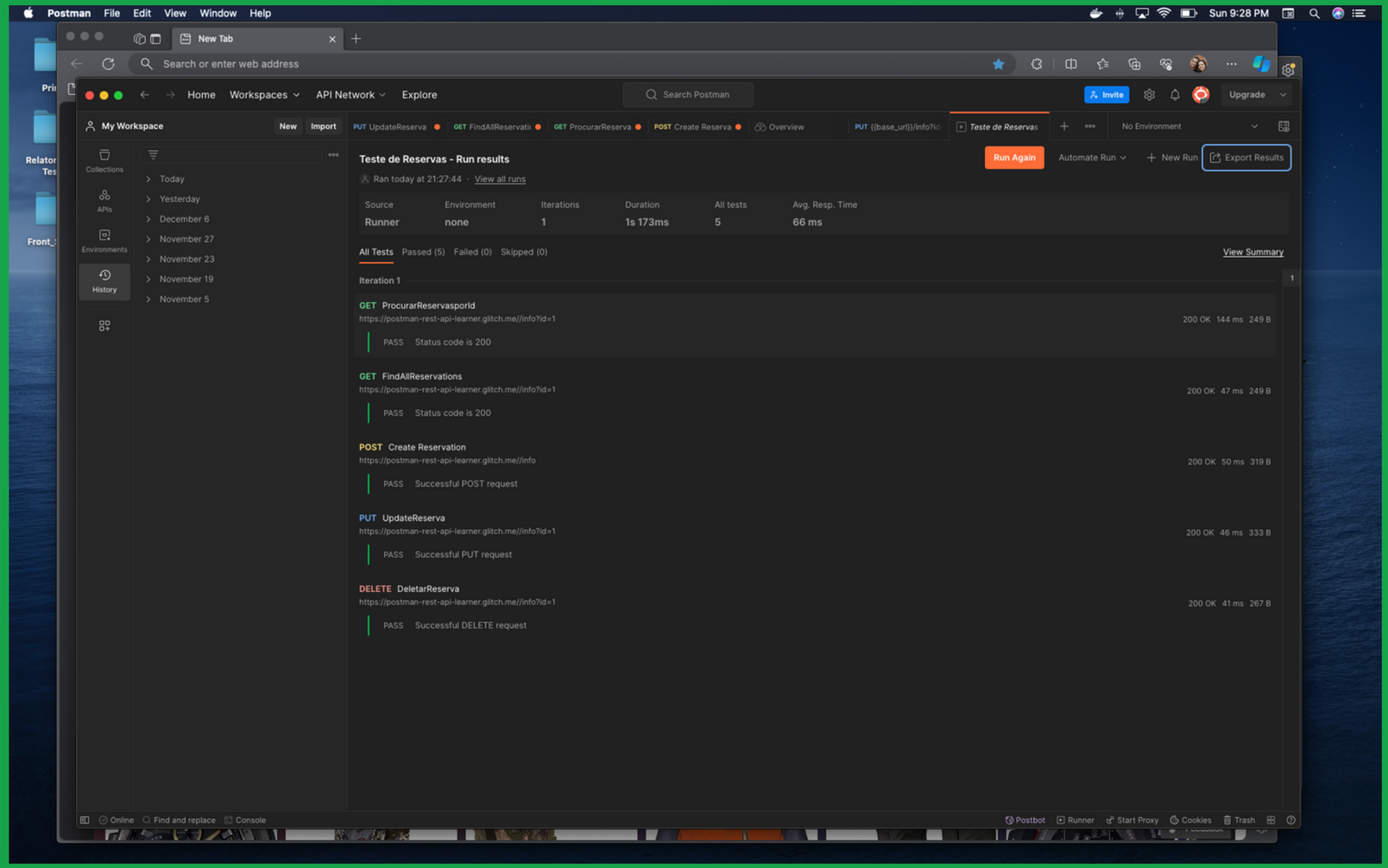


create.sql

```
    created_At DATETIME,
    updated_At DATETIME,
    host_contact_id BINARY(16),
    host_hotel_id BINARY(16),
    user_detail_id BINARY(16),
    FOREIGN KEY (host_contact_id) REFERENCES contact(id),
    FOREIGN KEY (host_hotel_id) REFERENCES hotel(id),
    FOREIGN KEY (user_detail_id) REFERENCES user_detail(id)
);



CREATE TABLE images (
    id BINARY(16) PRIMARY KEY,
    title VARCHAR(255),
    url VARCHAR(255) NOT NULL,
    hotel_id BINARY(16),
    FOREIGN KEY (hotel_id) REFERENCES hotel(id)
);


CREATE TABLE reservation (
    id BINARY(16) PRIMARY KEY,
    check_in_date DATE,
    check_out_date DATE,
    requests TEXT NOT NULL,
    is_canceled BOOLEAN DEFAULT false,
    reservation_start_time TIME,
    created_at DATETIME,
    updated_at DATETIME,
    isVaccinatedAgainstCovid BOOLEAN DEFAULT false,
    reservation_guest_id BINARY(16),
    reservation_host_id BINARY(16),
    FOREIGN KEY (reservation_guest_id) REFERENCES guest(id),
    FOREIGN KEY (reservation_host_id) REFERENCES host(id),
    INDEX (reservation_guest_id),
    INDEX (reservation_host_id)
```

TESTES

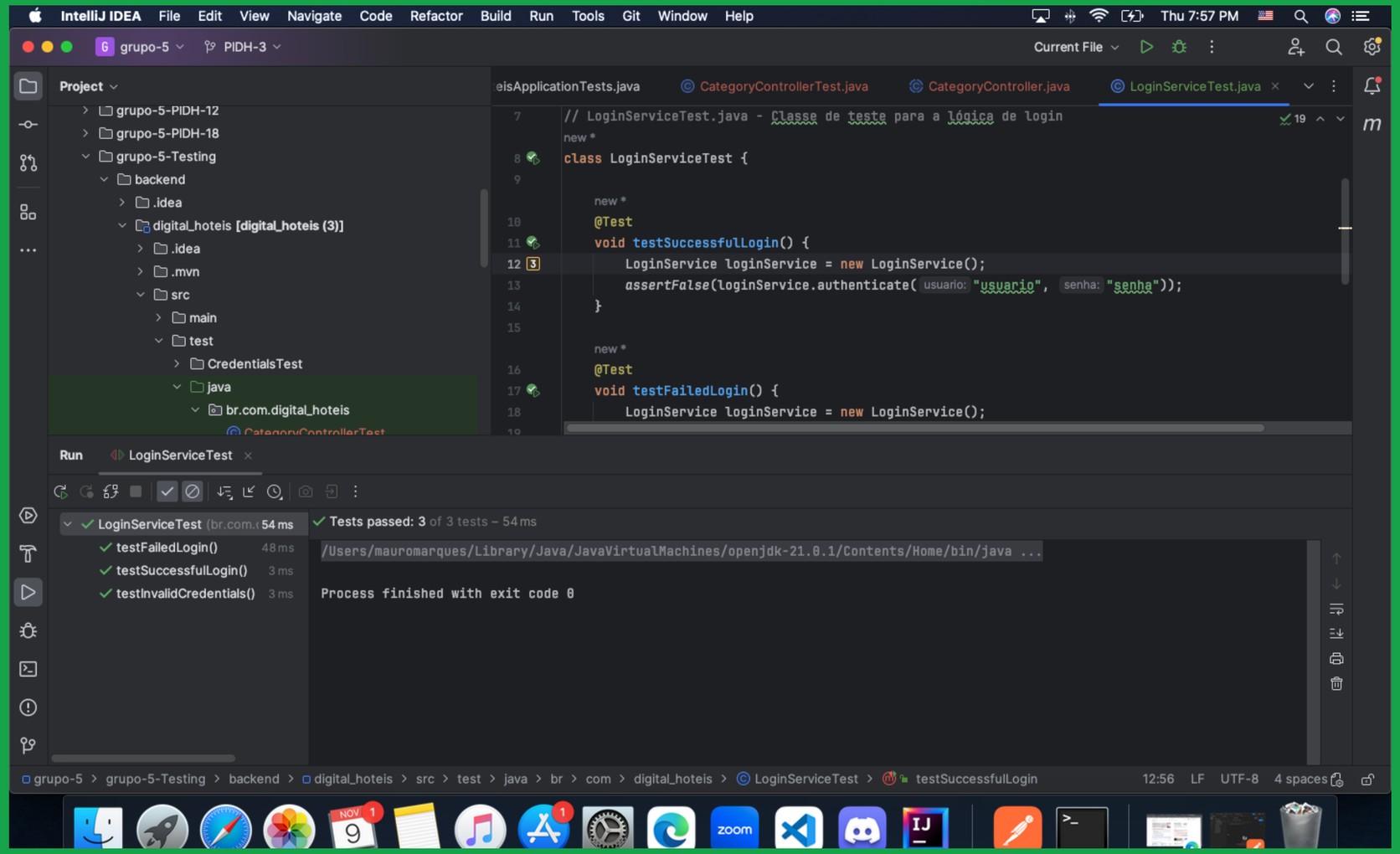
DETALHES DO QUE FEITO



Testes formaram 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 Auxilio da Patrícia Delvequi na formatação das ferramentas, o que possibilitou a execução devida dos testes.

TESTES

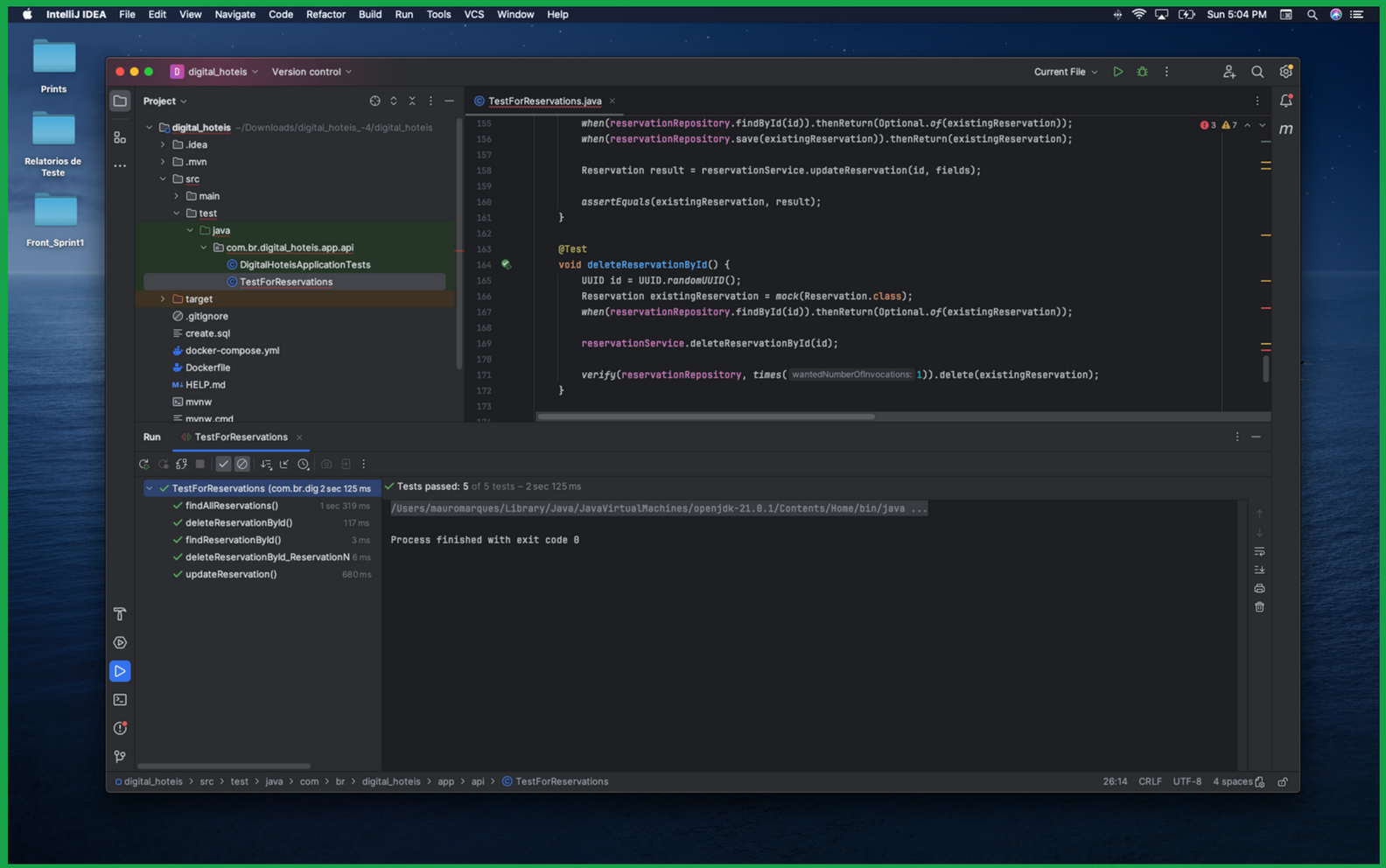
DETALHES DO QUE FEITO



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.

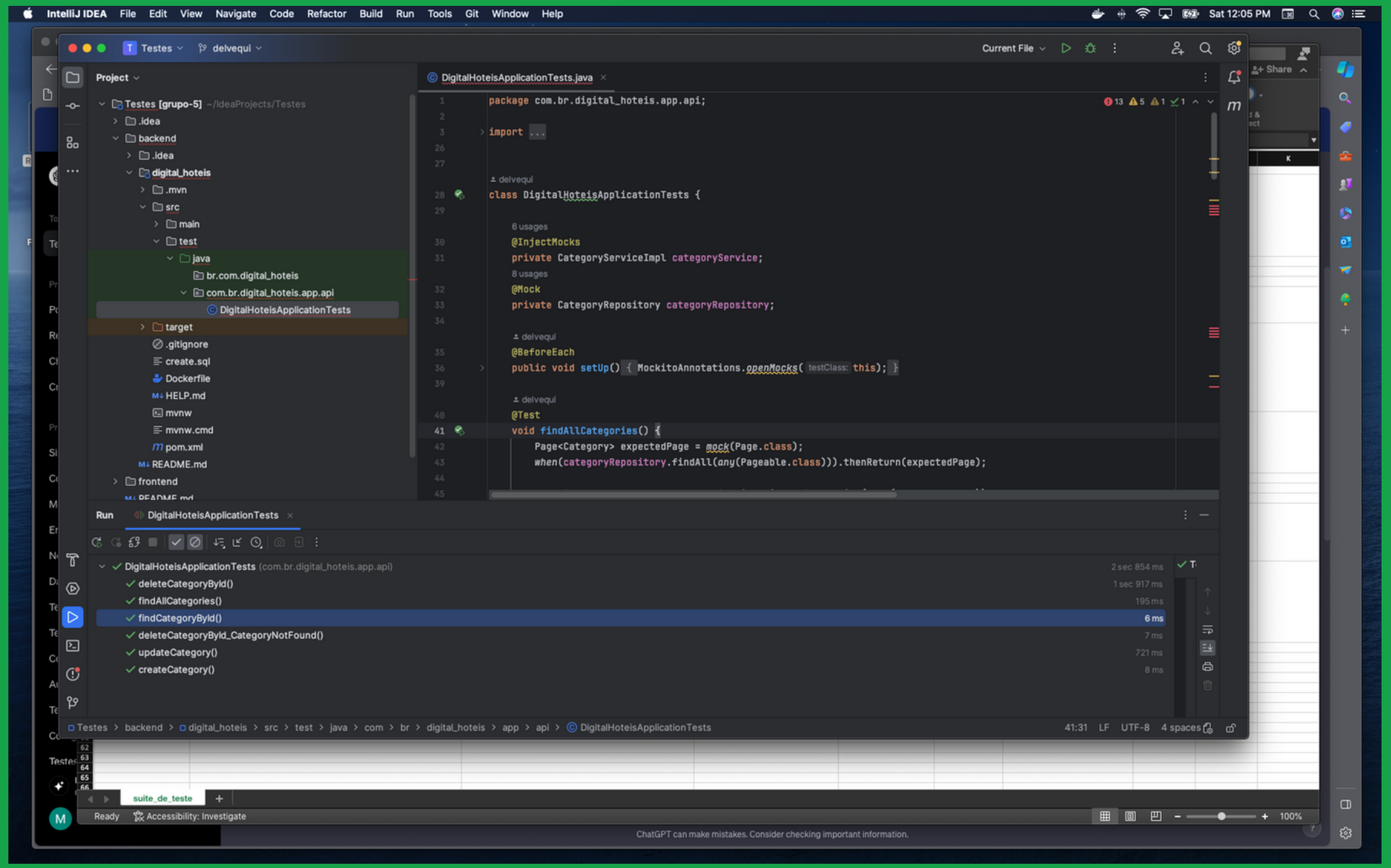
TESTES

DETALHES DO QUE FEITO



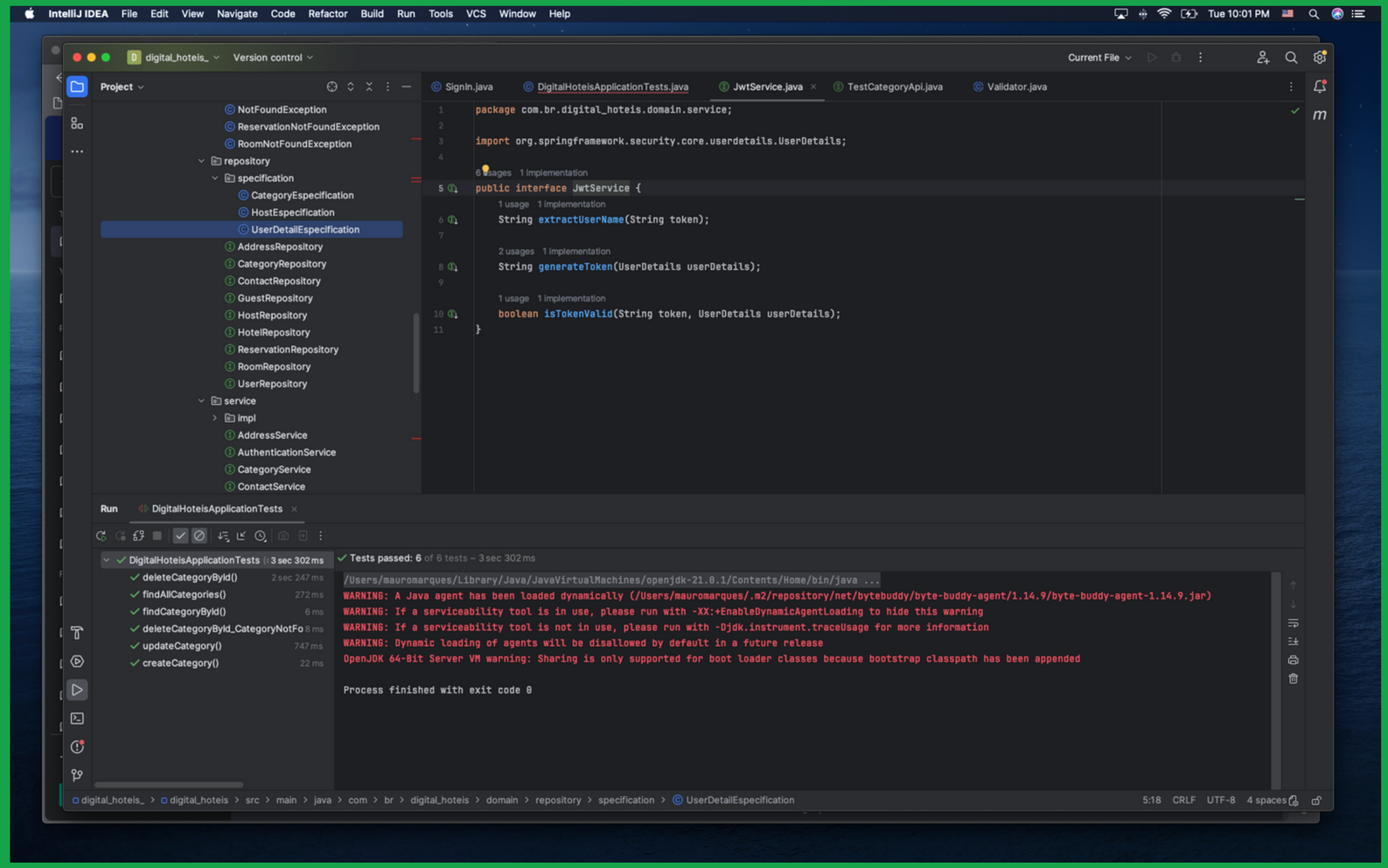
TESTES

DETALHES DO QUE FEITO



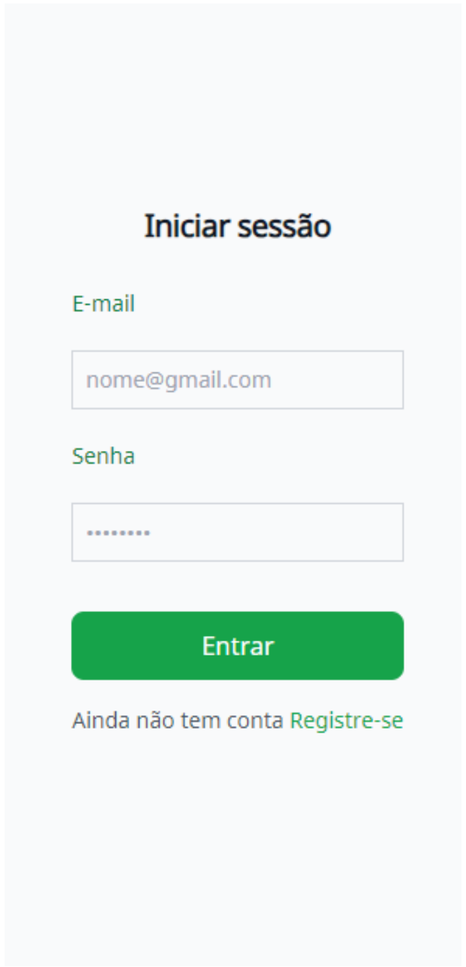
TESTES

DETALHES DO QUE FEITO



FRONT-END

DETALHES DO QUE FEITO



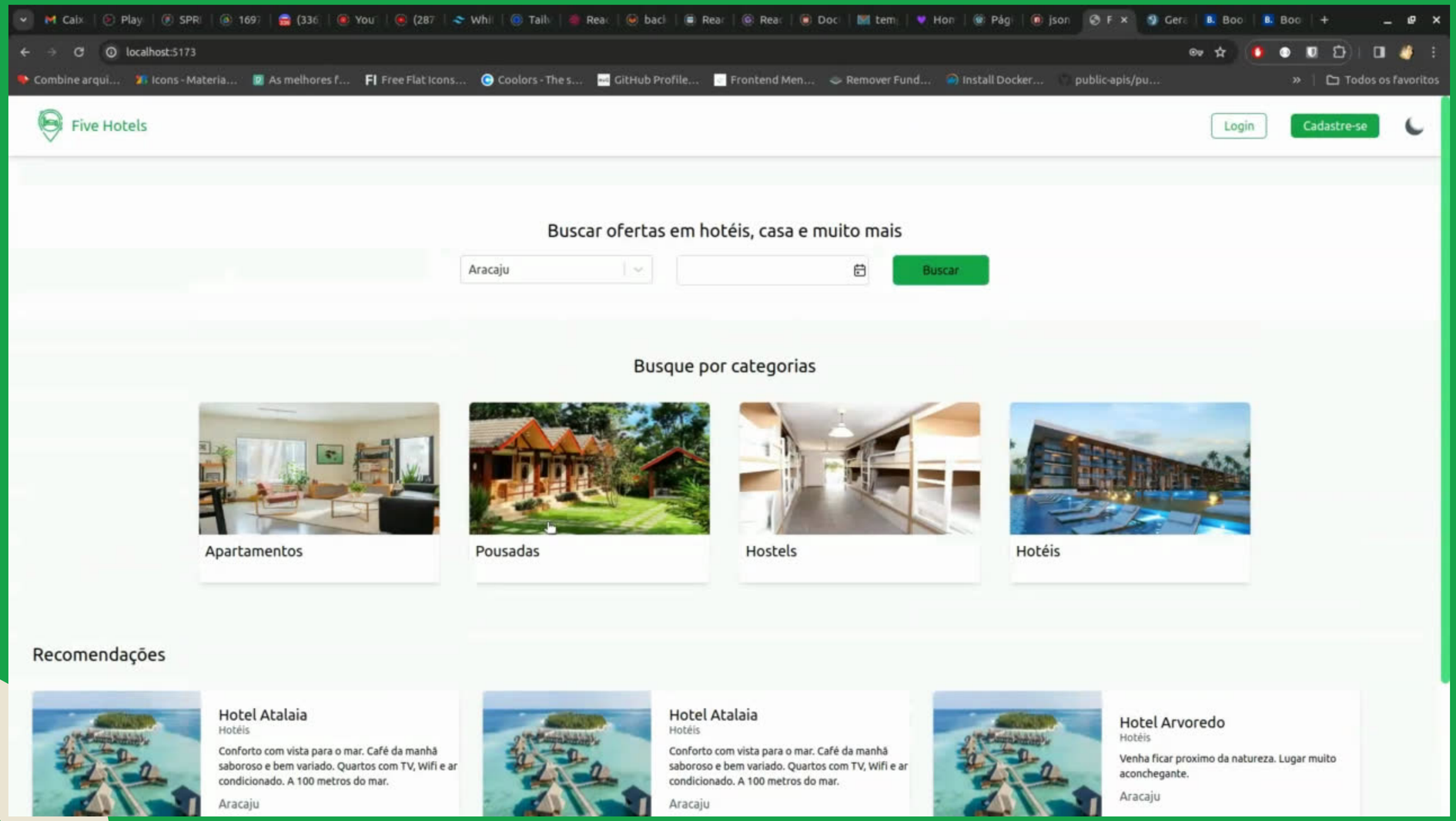
Login e Register

Administração

Detalhes dos Produtos

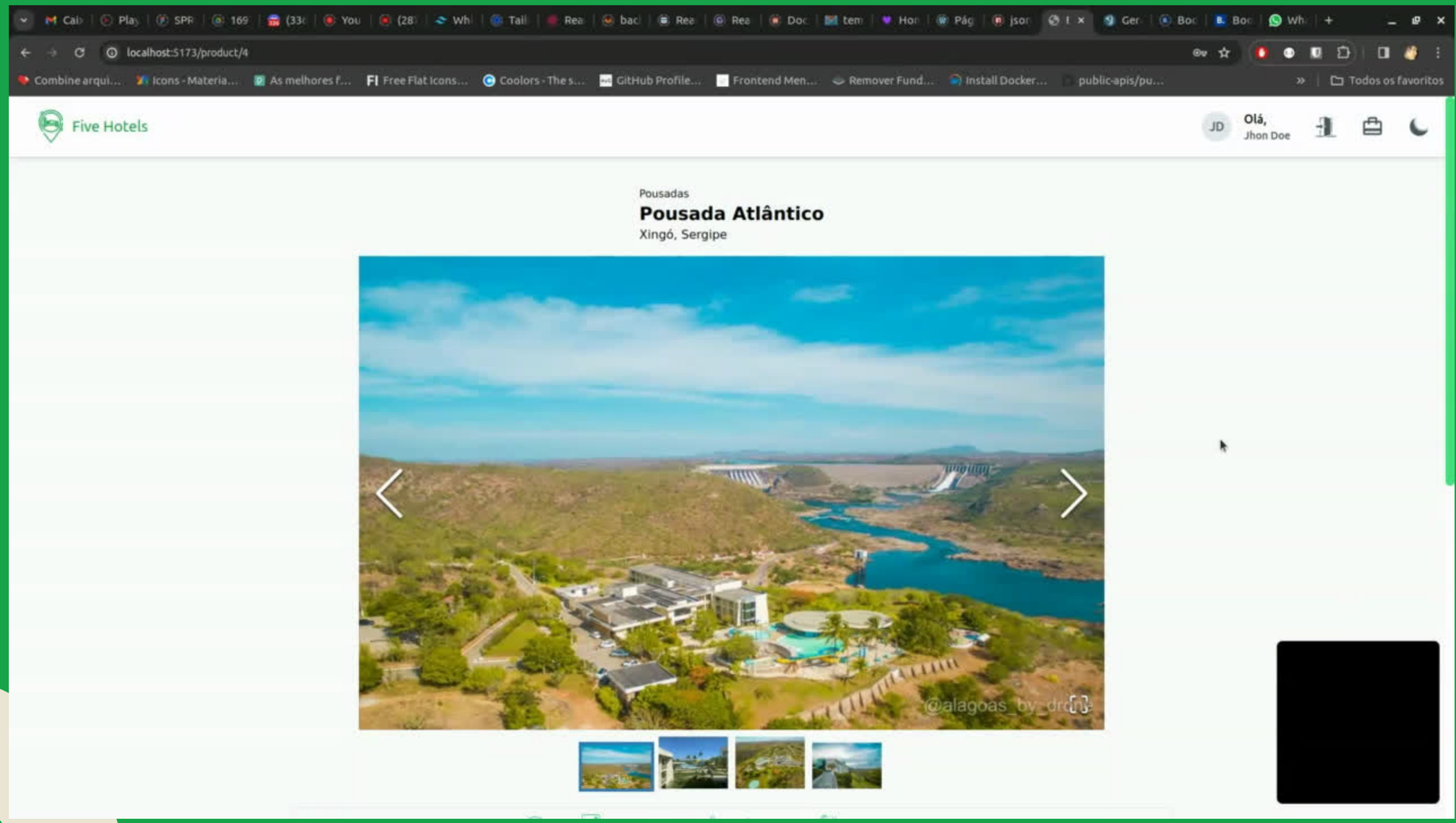
FRONT-END

DETALHES DO QUE FEITO



FRONT-END

DETALHES DO QUE FEITO



FRONT-END

DETALHES DO QUE FEITO