

# Rapport Mini Projet

## Système de Gestion de Réservation de Chambres d'Hôtel avec Spring Framework et React

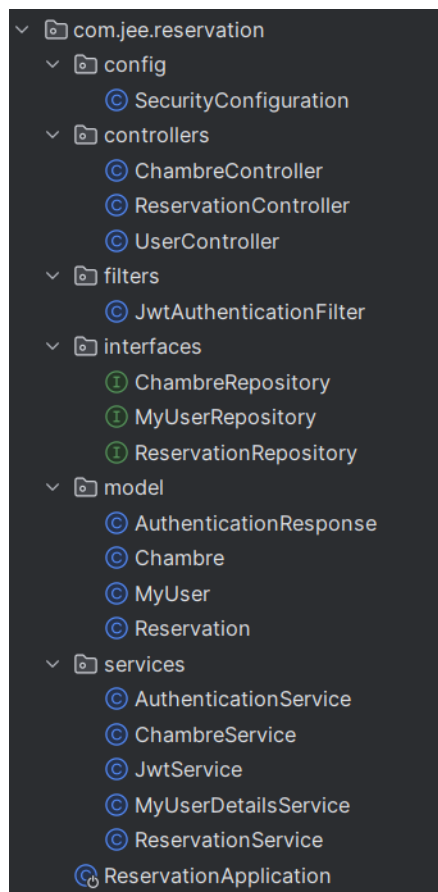
Seif Eddine Gherir – IGL 4

### I- Vue générale sur le projet

Le système de gestion de réservation de chambres d'hôtel permet aux clients de rechercher et réserver des chambres d'hôtel, et aux administrateurs de gérer les chambres, les réservations et les utilisateurs. Ce système est implémenté avec Spring au backend, et React au frontend.

Ceci se fait de manière sécurisée grâce à Spring Security et l'implémentation des tokens JWT. Toutes les données sont stockées dans une base de données grâce au Spring Data JPA.

Le projet utilise le modèle MVC (model-view-controller).



## II- Base de données

Une base de données MySQL est utilisée pour le stockage des données des réservations, des chambres ainsi que des utilisateurs.

**Table user(iduser, username, email, password, role)** : utilisée pour les données des utilisateurs. Le mot de passe de chacun est stocké crypté. Le champ rôle est soit USER si l'utilisateur est un client, soit ADMIN s'il est administrateur.

**Table chambre(idchambre, type, prix, disponibilite, description, image)** : utilisée pour les données des chambres. Le champ disponibilité est la date à partir de laquelle la chambre est disponible. Le champ image est l'URL d'une photo de la chambre.

**Table reservation(id\_reservation, date\_debut, date\_fin, idchambre#, iduser#, nombre\_personnes, status)** : idchambre et iduser sont les clés étrangères de la chambre et du client pour une réservation.

L'interaction du Spring avec la BDD se fait à travers Spring Data JPA. Ceci se fait en utilisant le mappage d'une classe en une table pour être stockés.

Par exemple : mappage de l'entité reservation :

```
32 usages
7  @Entity
8  @Table(name = "reservation")
9  public class Reservation {
10
11      @Id
12      @GeneratedValue(strategy = GenerationType.IDENTITY)
13      private int id_reservation;
14
15      2 usages
16      private LocalDate date_debut;
17      2 usages
18      private LocalDate date_fin;
19
20      2 usages
21      private int nombre_personnes;
22
23      2 usages
24      private String status;
25
26      2 usages
27      @ManyToOne
28      @JoinColumn(name = "idchambre")
29      private Chambre chambre;
30
31      2 usages
32      @ManyToOne
33      @JoinColumn(name = "iduser")
34      private MyUser user;
35 }
```

@Entity pour déclarer que c'est une entité

@Table pour spécifier à quelle table correspond cette entité

La clé primaire de la table est annotée avec @Id ,

@GeneratedValue est la stratégie de la création de l'id

Chaque attribut de la classe correspond à une colonne dans la table dans la BD.

@ManyToOne , pour la jointure entre différentes entités et @JoinColumn pour spécifier la colonne avec laquelle se fait la jointure.

### III- Fonctionnalités

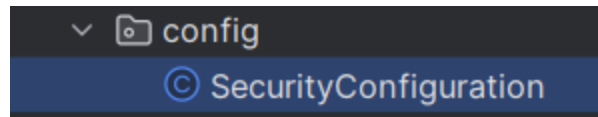
Pour afficher les différents APIs on utilise Swagger.

<b>user-controller</b>		^
GET	/users/{username}	▼
PUT	/users/{username}	▼
DELETE	/users/{username}	▼
PUT	/users/{username}/password	▼
POST	/register	▼
POST	/login	▼
GET	/users	▼
<b>reservation-controller</b>		^
GET	/reservation	▼
PUT	/reservation	▼
POST	/reservation	▼
GET	/reservation/{idReservation}	▼
DELETE	/reservation/{idReservation}	▼
GET	/reservation/guest/{username}	▼
<b>chambre-controller</b>		^
GET	/chambre/{id}	▼
PUT	/chambre/{id}	▼
DELETE	/chambre/{id}	▼
GET	/chambre	▼
POST	/chambre	▼
GET	/chambre/type/{type}	▼
GET	/chambre/disponible	▼
GET	/chambre/disponible/type/{type}	▼
GET	/chambre/disponible/type/{type}/date	▼
GET	/chambre/disponible/date	▼

## 1. Sécurité

Toutes les APIs sont sécurisées en utilisant Spring Security. Ce Framework est utilisé pour la gestion d'authentification des utilisateurs ainsi que le contrôle d'accès aux différentes fonctionnalités. Les tokens JWT sont utilisés pour renforcer la sécurité, surtout dans l'interaction du backend avec le frontend.

La configuration du Spring Security est spécifiée dans **SecurityConfiguration** du package **config**



Les autorisations des différentes fonctionnalités sont dans le Bean **securityFilterChain**.

```
@Bean
public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {...
```

Par exemple:

**authorizeRequests.requestMatchers(HttpMethod.POST, "/login/\*\*").permitAll();** -> toute requête POST vers l'URL **/login** est acceptée

**authorizeRequests.requestMatchers("/users/\*\*").hasAuthority("ADMIN");** -> toute requête vers l'URL **/users/\*\*** n'est acceptée que si l'utilisateur possède un token d'admin

Un filtre **jwtAuthenticationFilter** est implémenté aussi pour la vérification de la validité du token JWT

```
.addFilterBefore(jwtAuthenticationFilter, UsernamePasswordAuthenticationFilter.class)
```

```
@Override
protected void doFilterInternal(
    @NonNull HttpServletRequest request,
    @NonNull HttpServletResponse response,
    @NonNull FilterChain filterChain)
    throws ServletException, IOException {...
```

Ce filtre intercepte toute requête et récupère le header **Authorization** pour en extraire le token et configure ensuite l'authentification Spring Security après sa validation (définit l'authentification dans **SecurityContextHolder**).

La création et la validation du token se fait en utilisant **JwtService** du package **services**

## 2. Utilisateurs

### user-controller

GET	/users/{username}	^
PUT	/users/{username}	^
DELETE	/users/{username}	^
PUT	/users/{username}/password	^
POST	/register	^
POST	/login	^
GET	/users	^

**GET /users/{username}** : récupération d'un utilisateur à partir de son username

**PUT /users/{username}** : modification d'un utilisateur à partir de son username

**DELETE /users/{username}** : suppression d'un utilisateur à partir de son username

**PUT /users/{username}/password** : modification du mot de passe de l'utilisateur à partir de son username. Cette API est ajoutée pour que le mot de passe modifié soit stocké sous forme cryptée


**POST /register** : création d'un compte

**POST /login** : se connecter

**GET /users** : récupérer les informations sur tous les utilisateurs

L'utilisateur commence par la création d'un compte ou par se connecter en utilisant l'API POST sur /login ou /register





### Login

Username

Password

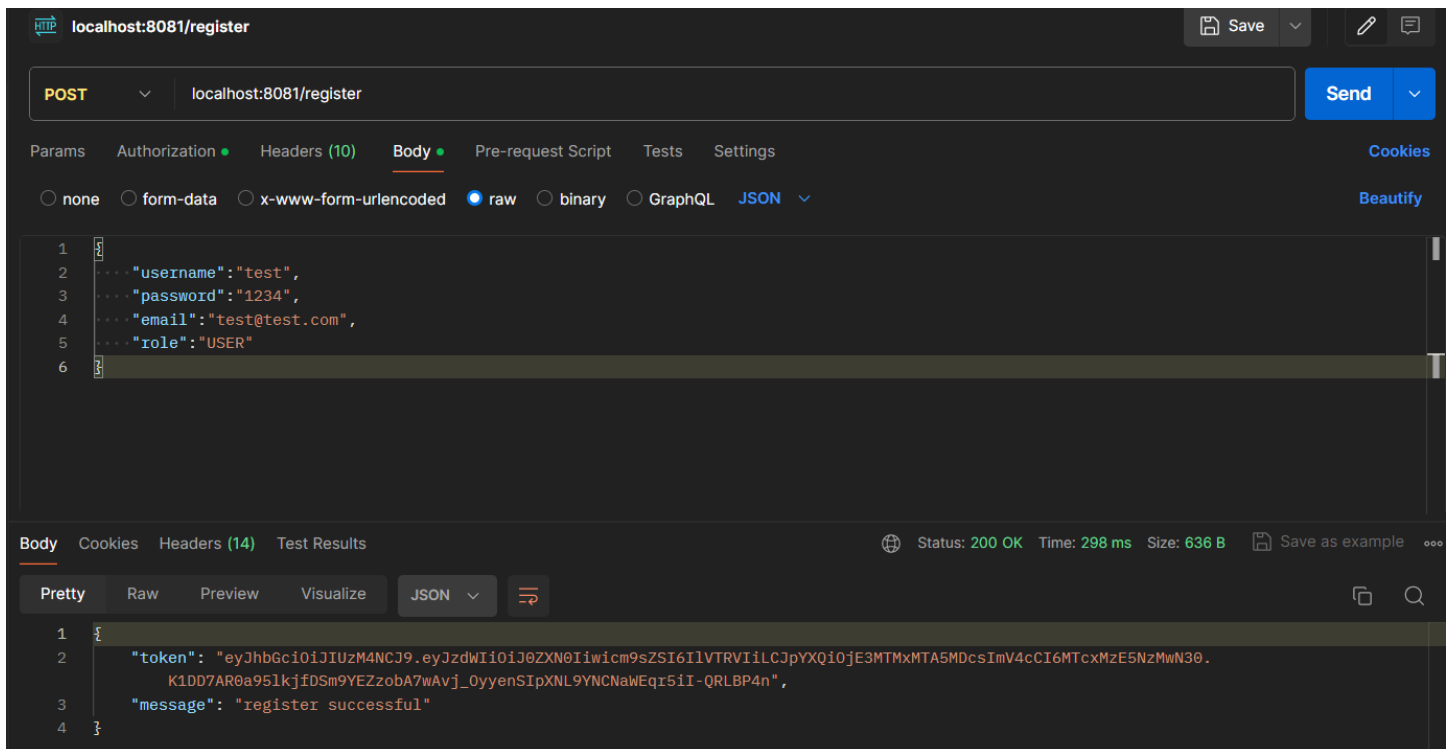
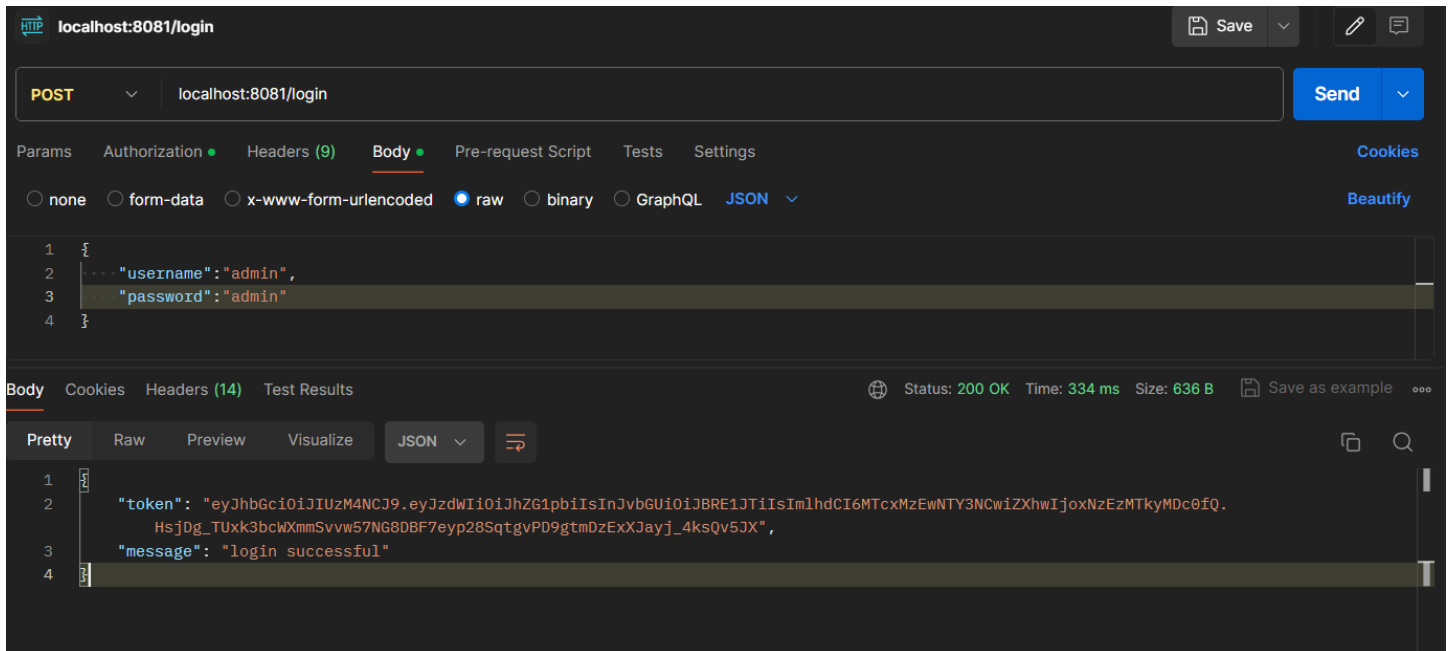
Login

Pas de compte? Créer un!

[Chambres](#) [Mes réservations](#) [Se connecter](#)



Si les coordonnées sont valides, un token est retourné.



iduser	username	email	password	role
17	test	test@test.com	\$2a\$10\$sc88xKc7yiXW9L9yHAWkmOfSDFF4RHUULeqtKLIX6e2...	USER

Ce token, ainsi que le username sont stockés dans le **localStorage** pour être utilisés dans le futur.

## L'utilisateur peut gérer son propre profile

[Chambres](#)[Mes réservations](#)[Modifier profile](#)[Déconnexion](#)

### Profile

Username

Email

Modifier le mot de passe

La modification se fait en faisant un **PUT /users/{username}** et nécessite son propre token. Essayons avec Postman.

Récupération d'abord du token de cet utilisateur

The screenshot shows a Postman interface for a POST request to `localhost:8081/login`. The request body is a JSON object: `{ "username": "test", "password": "1234" }`. The response status is `200 OK` with a time of `130 ms` and a size of `634 B`. The response body is a JSON object: `{ "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZW4iLCJ1eWUiOiJ0ZXN0Iiwicm9sZSI6IkpFETU0IiwiaWF0IjoxNzEzMTMxOTg5OTUwLCJleHAiOiJlM3MTMxOTg5OTUw", "message": "login successful" }`. The token is highlighted in the response body.

Faire un **PUT /users/{username}** en utilisant ce token

PUT

http://localhost:8081/users/test

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

Type

Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ5IiwiaWF0IjoxNjY0OTUwOTUwLCJleHAiOiJlbnR5bG9uZS50dGppABpayoRk2ZLVHb5aynG-xmuGW4Hxtxm8CrsBFoyNkfTgS3TKvDOWA0x6P

Body

Cookies

Headers (14)

Test Results

Status: 200 OK

Time: 51 ms

Size: 698 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

http://localhost:8081/users/test

Save

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

Body

Cookies

Headers (14)

Test Results

Status: 200 OK

Time: 51 ms

Size: 698 B

Save as example

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

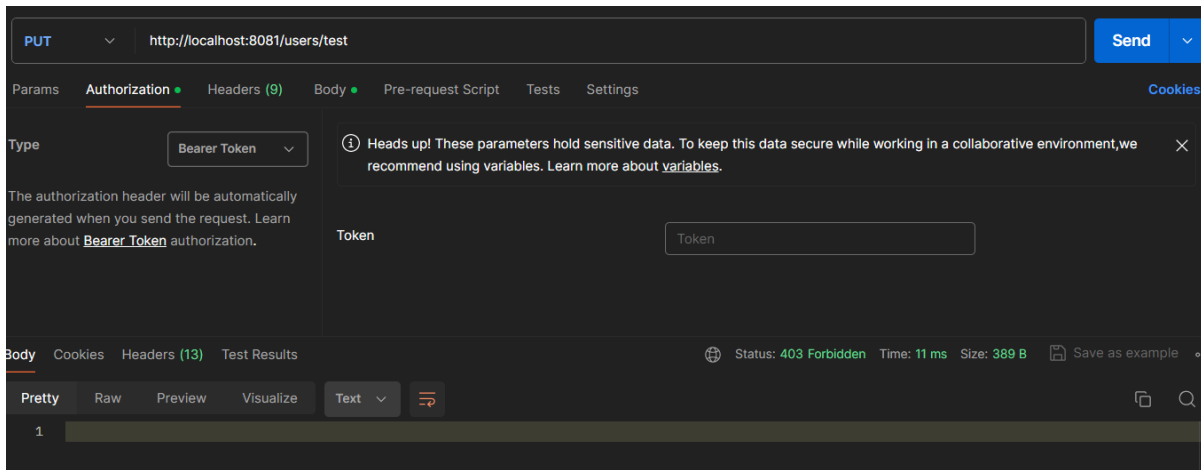
483

484

485



S'il n'y a pas de token spécifié, la réponse sera vide avec un statut **403 Forbidden**



Ceci est de même pour **PUT /users/{username}/password**

L'administrateur peut gérer les comptes d'utilisateurs.

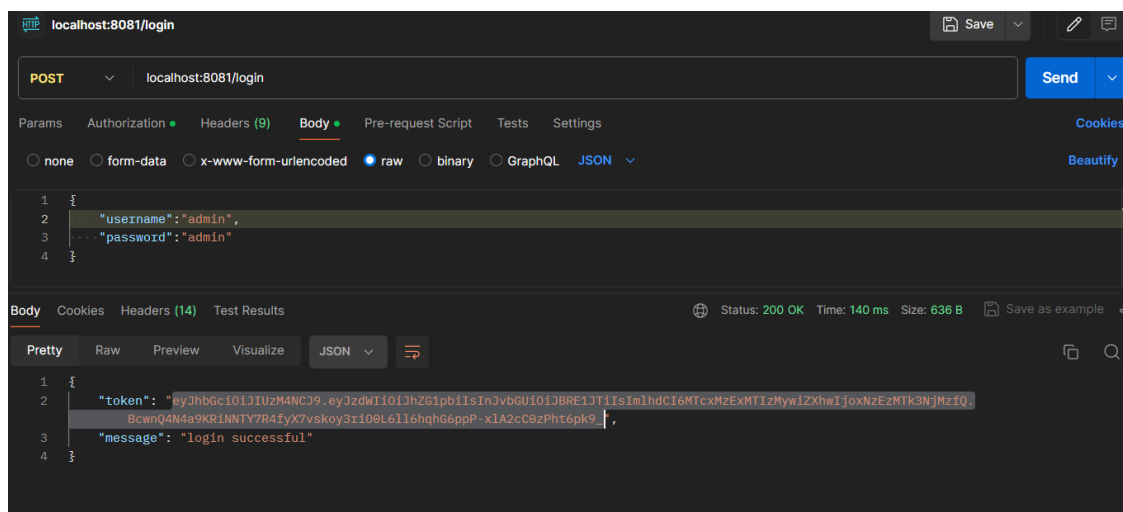


## Admin - Users

ID	Username	Email	Role	Actions	
17	test	test@mail11.com	USER	<button>Edit</button>	<button>Delete</button>
6	admin	seef@swif.com	ADMIN	<button>Edit</button>	<button>Delete</button>
9	seef	seef@seef.com	USER	<button>Edit</button>	<button>Delete</button>

Ceci se fait en faisant un **GET** sur **/users** d'abord pour charger tous les utilisateurs. Essayons ceci sur Postman.

Il faut d'abord se connecter comme admin pour récupérer son token



Puis utiliser ce token pour faire le **GET**.

The screenshot shows a Postman interface for a GET request to `http://localhost:8081/users`. The request is configured with a Bearer Token. The response status is 200 OK, and the body is a JSON object representing a user.

**Request:**

- Method: GET
- URL: `http://localhost:8081/users`
- Authorization: Bearer Token

**Response Body (JSON):**

```
[{"iduser": 17, "username": "test", "email": "test@test.com", "password": "$2a$10$sc88xKc7yiXw9L9yHAWkm0fSDFF4RHUULeqtKL1X6e2uvCVH0F0dS", "role": "USER", "enabled": true, "accountNonLocked": true, "authorities": [{"authority": "USER"}]}
```

Si on fait un **GET** avec le token d'un utilisateur avec le rôle USER, la réponse sera vide avec un statut **403 Forbidden**.

The screenshot shows a Postman interface for a GET request to `http://localhost:8081/users`. The request is configured with a Bearer Token. The response status is 403 Forbidden, and the body is empty.

**Request:**

- Method: GET
- URL: `http://localhost:8081/users`
- Authorization: Bearer Token

**Response:**

- Status: 403 Forbidden
- Time: 17 ms
- Size: 389 B

L'administrateur peut aussi changer les coordonnées et le rôle de l'utilisateur avec **PUT /users/{username}**

ID	Username	Email	Role	Actions
17	test	test@mail.com	<input type="text" value="ADMIN"/>	<button>Save</button> <button>Delete</button>

Equivalent avec Postman :

The screenshot shows the Postman interface for a PUT request to `http://localhost:8081/users/test`. The request body is a JSON object with the following fields:

```
{  "iduser": 17,  "username": "test",  "email": "test@mail.com",  "password": "$2a$10$sc88xKc7yiXW9L9yHAWkm0fSDFF4RHUULeqtKLlX6e2uvCVH0F0dS",  "role": "ADMIN"}
```

The response status is **200 OK**, with a time of 21 ms and a size of 698 B. The response body is a JSON object with the following fields:

```
{  "iduser": 17,  "username": "test",  "email": "test@mail.com",  "password": "$2a$10$sc88xKc7yiXW9L9yHAWkm0fSDFF4RHUULeqtKLlX6e2uvCVH0F0dS",  "role": "ADMIN",  "enabled": true,  "accountNonLocked": true,  "authorities": [  ]}
```

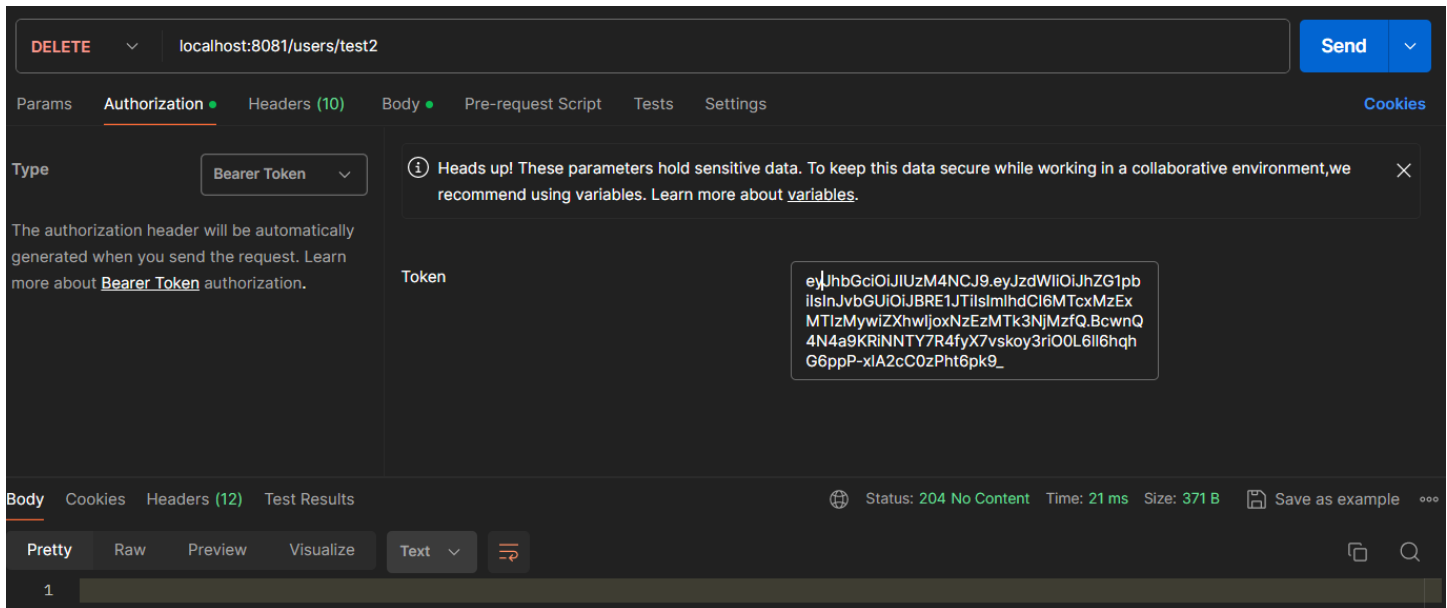
Si on essaie de faire un **PUT** avec le token d'un utilisateur être que l'utilisateur concerné et l'administrateur, la réponse sera **403 Forbidden**

The screenshot shows the Postman interface for a PUT request to `http://localhost:8081/users/test`. The request is authorized with a Bearer Token. The response status is **403 Forbidden**, with a time of 12 ms and a size of 389 B. The response body is a text message:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXZWVmaWwMTQslmV4cCI6MTcxMzE5ODQxNH0uL5k8lCTzmSaPaGzMTIe8clvBqAv9V6xkNrjxoQzl6FjrPQHbFeoJ2c7NcB1fstY
```

L'administrateur peut faire un **DELETE /users/{username}** pour supprimer un utilisateur

18	test2	test	USER	<button>Confirm</button>
----	-------	------	------	--------------------------



### Implémentation :

Le modèle utilisé est **MyUser** dans le package **models** et contient le mappage par rapport à la table dans la base de données.

Le logique métier concernant les utilisateurs se trouve dans le service **MyUserDetailsService** qui implémente **UserDetailsService** (pour la configuration de Spring Security) et qui utilise l'interface **MyUserRepository**.

```
10 usages
public interface MyUserRepository extends JpaRepository<MyUser, Integer> {

    3 usages
    Optional<MyUser> findByUsername(String username); // optional: dans le cas où l'utilisateur n'existe pas

    5 usages
    MyUser getUserByUsername(String username);

    1 usage
    void deleteByUsername(String username);
}
```

**MyUserRepository** étend **JpaRepository** qui est une interface fournie par **Spring Data JPA** qui étend l'interface **CrudRepository**. Cette interface fournit des méthodes prédéfinies pour effectuer des opérations CRUD sur une entité JPA.

Le service **MyUserDetailsService** prend **MyUserRepository** comme attribut, et il est annoté avec **@Service**

```
@Service
public class MyUserDetailsService implements UserDetailsService {

    @Autowired
    private MyUserRepository repository;

    public MyUserDetailsService(MyUserRepository repository) { this.repository = repository; }

    1 usage
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {...}
}
```

Les services seront exposés avec le contrôleur **UserController** dans le package **controllers**.

Le contrôleur est annoté avec **@RestController**

**@CrossOrigin(\*)** est ajouté pour que le frontend accède au backend.

Chaque service est annoté avec soit **@GetMapping**, **@PostMapping**, **@DeleteMapping** ou **@PutMapping** pour spécifier la méthode utilisée ainsi que son URL.

```

@CrossOrigin("*")
@RestController
public class UserController {

    3 usages
    private final AuthenticationService authService;

    @Autowired
    private MyUserRepository userRepository;

    @Autowired
    private ReservationService reservationService;

    @Autowired
    private PasswordEncoder passwordEncoder;

    public UserController(AuthenticationService authService) { this.authService = authService; }

    @PostMapping("/register")
    public ResponseEntity<AuthenticationResponse> register(@RequestBody MyUser request) {...}

    @PostMapping("/login")
    public ResponseEntity<AuthenticationResponse> login(@RequestBody MyUser request) {...}

    @GetMapping("/users/{username}")
    public ResponseEntity<MyUser> getUser(@PathVariable String username) {...}

    @DeleteMapping("/users/{username}")
    @Transactional
    public ResponseEntity<Void> deleteUser(@PathVariable String username) {...}

    @PutMapping("/users/{username}")
    public ResponseEntity<MyUser> updateUser(@PathVariable String username, @RequestBody MyUser user) {...}

    @PutMapping("/users/{username}/password")
    public ResponseEntity<?> changePassword(@PathVariable String username, @RequestBody String newPasswordJSON)
        throws JsonProcessingException {...}

    @GetMapping("/users")
    public ResponseEntity<Iterable<MyUser>> getUsers() {...}

```

### 3. Chambres

chambre-controller		^
GET	/chambre/{id}	▼
PUT	/chambre/{id}	▼
DELETE	/chambre/{id}	▼
GET	/chambre	▼
POST	/chambre	▼
GET	/chambre/type/{type}	▼
GET	/chambre/disponible	▼
GET	/chambre/disponible/type/{type}	▼
GET	/chambre/disponible/type/{type}/date	▼
GET	/chambre/disponible/date	▼

**GET /chambre/{id}** : récupérer une chambre par son ID

**PUT /chambre/{id}** : modifier une chambre par son ID

**DELETE /chambre/{id}** : supprimer une chambre par son ID

**POST /chambre** : ajouter une chambre

**GET /chambre** : récupérer toutes les chambres

**Filtres :**

**GET /chambre/type/{type}** : récupérer les chambres par type

**GET /chambre/disponible** : récupérer les chambres disponibles pour la date actuelle

**GET /chambre/disponible/type/{type}** : récupérer les chambres disponibles pour la date actuelle par type

**GET /chambre/disponible/date** : récupérer les chambres disponibles par date de départ et date d'arrivée

**GET /chambre/disponible/type/{type}/date** : récupérer les chambres disponibles par date de départ et date d'arrivée par type



## Chambres

### Filtrer les chambres

Veuillez sélectionner au moins un filtre autre que le type.

Type :

Tous les types

☐ Disponibles maintenant

Date de début :

01/01/2025



Date de fin :

01/02/2026



Filtrer



Simple  
13

Chambre simple avec lit  
simple avec vue sur mer  
699 €

Plus de détails



Appartement  
11

Appartement spacieux avec  
cuisine et salon  
900 €

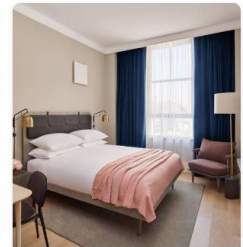
Plus de détails



Simple  
14

Chambre simple avec lit  
simple  
210 €

Plus de détails



Simple  
16

Chambre simple avec lit  
simple avec vue sur mer  
210 €

Plus de détails

localhost:3000/chambre/13

Les chambres sont affichées en utilisant l'API **GET /chambre**, les chambres seront filtrées en utilisant les différents filtres spécifiés précédemment.

Test avec Postman des différents filtres:



http://localhost:8081/chambre

Save

GET

http://localhost:8081/chambre

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

BodyCookiesHeaders (14)Test Results

Status: 200 OK

Time: 35 ms

Size: 3.76 KB

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "idchambre": 13,
4     "type": "Simple",
5     "prix": 699,
6     "disponibilite": "2024-04-17",
7     "description": "Chambre simple avec lit simple avec vue sur mer",
8     "image": "https://i.pinimg.com/736x/4e/ef/8c/4eef8c8ce0f8d2bc97c43a7d45e8b875.jpg"
9   },
10  {
11    "idchambre": 9,
12    "type": "Double",
13    "prix": 304,
14    "disponibilite": "2024-04-13",
15    "description": "Chambre double luxueuse avec vue sur mer",
16    "image": "https://i.pinimg.com/736x/93/4d/b7/934db7616c51a235d580c2e7fd589bdc.jpg"
17  },
18  {
19    "idchambre": 10,
20    "type": "Suite",
21    "prix": 460,
22    "disponibilite": "2029-01-04",
```

http://localhost:8081/chambre/type/Penthouse

Save

GET

http://localhost:8081/chambre/type/Penthouse

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

BodyCookiesHeaders (14)Test Results

Status: 200 OK

Time: 8 ms

Size: 1.04 KB

Save as example

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "idchambre": 12,
4     "type": "Penthouse",
5     "prix": 1200,
6     "disponibilite": "2027-01-02",
7     "description": "Penthouse de luxe avec vue panoramique",
8     "image": "https://i.pinimg.com/736x/a3/ac/3c/a3ac3c7886ded2fbfa0ae817f940dcf6.jpg"
9   },
10  {
11    "idchambre": 21,
12    "type": "Penthouse",
13    "prix": 1200,
14    "disponibilite": "2024-04-13",
15    "description": "Penthouse de luxe avec vue panoramique",
16    "image": "https://i.pinimg.com/736x/a3/ac/3c/a3ac3c7886ded2fbfa0ae817f940dcf6.jpg"
17  },
18  {
```

HTTP

http://localhost:8081/chambre/disponible/date?dateArrivee=2025-01-01&dateDepart=2026-01-02

Save

GET

http://localhost:8081/chambre/disponible/date?dateArrivee=2025-01-01&dateDepart=2026-01-02

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

BodyCookiesHeaders (14)Test Results

Status: 200 OKTime: 49 msSize: 3.14 KBSave as example

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "idchambre": 13,
4     "type": "Simple",
5     "prix": 699,
6     "disponibilite": "2024-04-17",
7     "description": "Chambre simple avec lit simple avec vue sur mer",
8     "image": "https://i.pinimg.com/736x/4e/ef/8c/4eef8c8ce0fd2bc97c43a7d45e8b875.jpg"
9   },
10  {
11    "idchambre": 11,
12    "type": "Appartement",
13    "prix": 900,
14    "disponibilite": "2024-04-13",
15    "description": "Appartement spacieux avec cuisine et salon",
16    "image": "https://i.pinimg.com/736x/77/64/72/7764727393a444b8c0fc30886275e1d8.jpg"
17  },
18  {
19    "idchambre": 14,
20    "type": "Simple",
```

HTTP

http://localhost:8081/chambre/disponible

Save

GET

http://localhost:8081/chambre/disponible

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

BodyCookiesHeaders (14)Test Results

Status: 200 OKTime: 36 msSize: 3.13 KBSave as example

PrettyRawPreviewVisualizeJSON

```
1 [
2   {
3     "idchambre": 9,
4     "type": "Double",
5     "prix": 304,
6     "disponibilite": "2024-04-13",
7     "description": "Chambre double luxueuse avec vue sur mer",
8     "image": "https://i.pinimg.com/736x/93/4d/b7/934db7616c51a235d580c2e7fd589bdc.jpg"
9   },
10  {
11    "idchambre": 11,
12    "type": "Appartement",
13    "prix": 900,
14    "disponibilite": "2024-04-13",
15    "description": "Appartement spacieux avec cuisine et salon",
16    "image": "https://i.pinimg.com/736x/77/64/72/7764727393a444b8c0fc30886275e1d8.jpg"
17  },
18  {
19    "idchambre": 14,
```

GET <http://localhost:8081/chambre/disponible/type/Appartement/date?dateArrivee=2025-01-01&dateDepart=2026-01-02> Send

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (14) Test Results Status: 200 OK Time: 24 ms Size: 862 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "idchambre": 11,
4     "type": "Appartement",
5     "prix": 900,
6     "disponibilite": "2024-04-13",
7     "description": "Appartement spacieux avec cuisine et salon",
8     "image": "https://i.pinimg.com/736x/77/64/72/7764727393a444b8c0fc30886275e1d8.jpg"
9   },
10  {
11    "idchambre": 20,
12    "type": "Appartement",
13    "prix": 900,
14    "disponibilite": "2024-04-13",
15    "description": "Appartement spacieux avec cuisine et salon",
16    "image": "https://i.pinimg.com/736x/77/64/72/7764727393a444b8c0fc30886275e1d8.jpg"
17  }
18 ]
```

Pour afficher les détails d'une chambre spécifique, on utilise **GET /chambre/{id}**



Chambres Mes réservations Modifier profile Déconnexion



## Suite

Suite luxueuse avec salon et chambre à coucher

### Équipements :

Connexion Wi-Fi gratuite  
Télévision à écran plat  
Climatisation réglable  
Salle de bains privative  
Service en chambre  
Espace de travail confortable  
Cafetière et théière

Prix: 460 €

Disponible

Réserver

HTTP <http://localhost:8081/chambre/10> Save

GET <http://localhost:8081/chambre/10> Send

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (14) Test Results Status: 200 OK Time: 14 ms Size: 640 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "idchambre": 10,
3   "type": "Suite",
4   "prix": 460,
5   "disponibilite": "2029-01-04",
6   "description": "Suite luxueuse avec salon et chambre à coucher",
7   "image": "https://i.pinimg.com/736x/e2/84/dc/e284dc03c0246b64033fcf33e29b4679.jpg"
8 }
```

L'administrateur peut ajouter, modifier et supprimer les chambres. Ces API sont sécurisées (nécessitent un token d'un admin seulement)



[Chambres](#) [Mes réservations](#) [Modifier profile](#) [Déconnexion](#)

## Admin - Chambres

Type	Description	Prix	Disponibilité	Image URL	Add Chambre	
ID	Type	Description	Prix	Disponibilité	Actions	
13	Simple	Chambre simple avec lit simple avec vue sur mer	699	2024-04-17	<a href="#">Edit</a>	<a href="#">Delete</a>
9	Double	Chambre double luxueuse avec vue sur mer	304	2024-04-13	<a href="#">Edit</a>	<a href="#">Delete</a>
10	Suite	Suite luxueuse avec salon et chambre à coucher	460	2029-01-04	<a href="#">Edit</a>	<a href="#">Delete</a>
11	Appartement	Appartement spacieux avec cuisine et salon	900	2024-04-13	<a href="#">Edit</a>	<a href="#">Delete</a>
12	Penthouse	Penthouse de luxe avec vue panoramique	1200	2027-01-02	<a href="#">Edit</a>	<a href="#">Delete</a>
14	Simple	Chambre simple avec lit simple	210	2024-04-13	<a href="#">Edit</a>	<a href="#">Delete</a>
16	Simple	Chambre simple avec lit simple avec vue sur mer	210	2024-04-13	<a href="#">Edit</a>	<a href="#">Delete</a>
18	Double	Chambre double luxueuse avec vue sur mer	304	2024-04-13	<a href="#">Edit</a>	<a href="#">Delete</a>
19	Suite	Suite luxueuse avec salon et chambre à coucher	460	2024-04-13	<a href="#">Edit</a>	<a href="#">Delete</a>

## Modification :

PUT

http://localhost:8081/chambre/13

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "idchambre": 13,
3   "type": "Simple",
4   "prix": 250,
5   "disponibilite": "2024-04-17",
6   "description": "Chambre simple avec lit simple avec vue sur mer",
7   "image": "https://i.pinimg.com/736x/4e/ef/8c/4eef8c8ce0f8d2bc97c43a7d45e8b875.jpg"
8 }
```

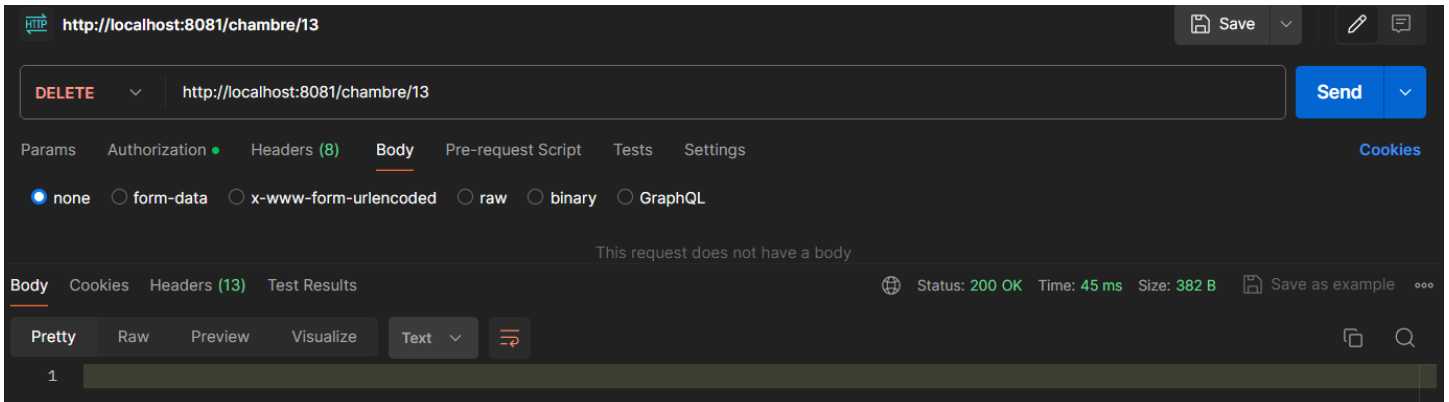
Body Cookies Headers (14) Test Results Status: 200 OK Time: 22 ms Size: 641 B Save as example

Pretty Raw Preview Visualize JSON

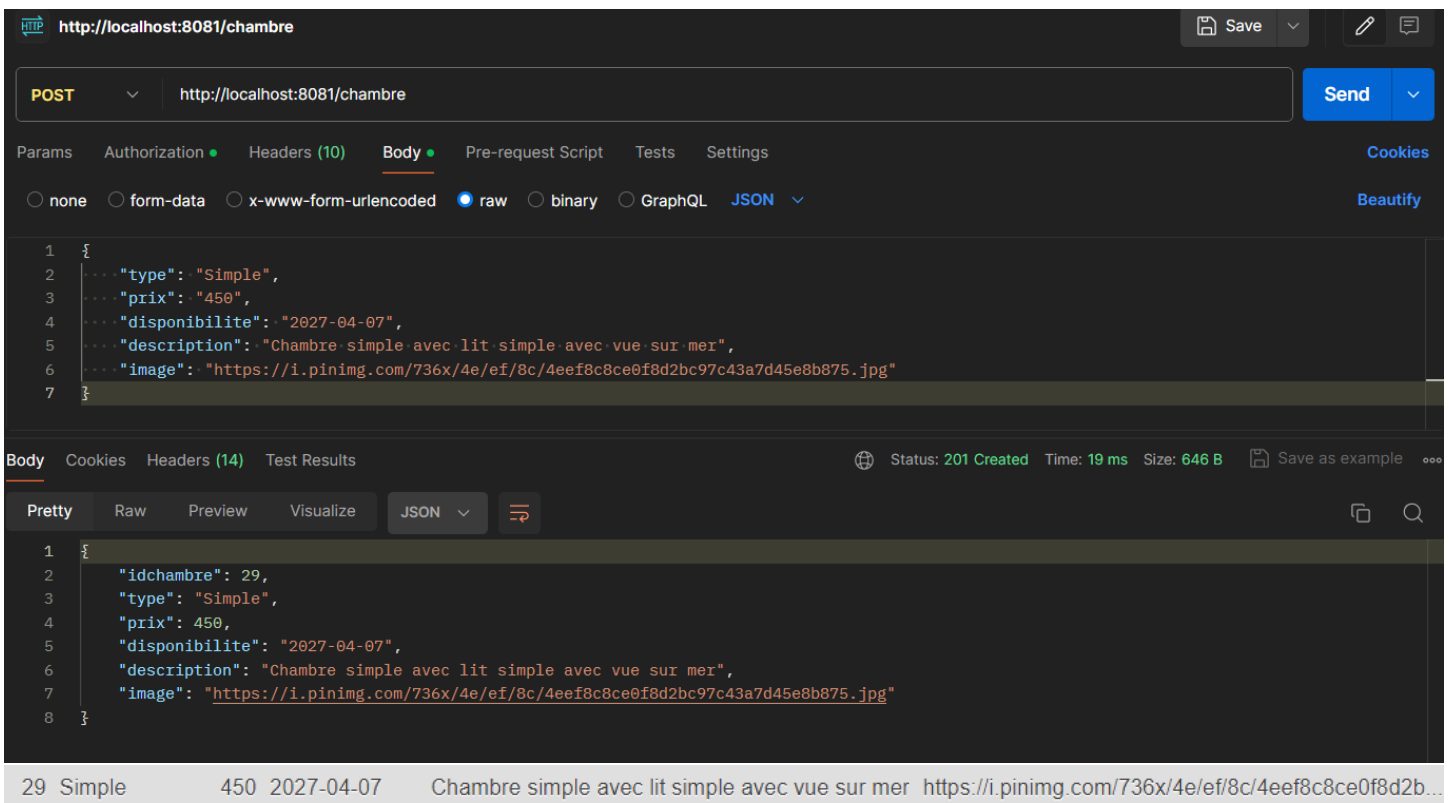
```
1 {
2   "idchambre": 13,
3   "type": "Simple",
4   "prix": 250,
5   "disponibilite": "2024-04-17",
6   "description": "Chambre simple avec lit simple avec vue sur mer",
7   "image": "https://i.pinimg.com/736x/4e/ef/8c/4eef8c8ce0f8d2bc97c43a7d45e8b875.jpg"
8 }
```

idchambre	type	prix	disponibilite	description	image
13	Simple	250	2024-04-17	Chambre simple avec lit simple avec vue sur mer	https://i.pinimg.com/736x/4e/ef/8c/4eef8c8ce0f8d2b...

Suppression :



Ajout :



Implémentation :

Le modèle est **Chambre** dans le package **models**

Le logique métier est dans **ChambreService** qui utilise **ChambreRepository** qui étend **JpaRepository**

```
4 usages
public interface ChambreRepository extends JpaRepository<Chambre, Integer> {
    1 usage
    List<Chambre> findAllByDisponibiliteBefore(LocalDate disponibilite);

    1 usage
    List<Chambre> findAllByIdchambreIsNotIn(List<Integer> chambreIds);

    1 usage
    List<Chambre> findAllByType(String type);|
```

```
2 usages
@Service
public class ChambreService {

    @Autowired
    private ChambreRepository chambreRepository;

    @Autowired
    private ReservationRepository reservationRepository;

    1 usage
    public Chambre createChambre(Chambre chambre) { return chambreRepository.save(chambre); }

    1 usage
    public Chambre getChambreById(Integer id) { return chambreRepository.findById(id).orElse( other: null); }

    1 usage
    public Chambre updateChambre(Chambre chambre) { return chambreRepository.save(chambre); }

    1 usage
    public void deleteChambre(Integer id) { chambreRepository.deleteById(id); }

    1 usage
    public List<Chambre> getAllChambres() { return chambreRepository.findAll(); }

    2 usages
    public List<Chambre> getAllAvailableChambres() {
        return chambreRepository.findAllByDisponibiliteBefore(LocalDate.now());
    }

    1 usage
    public List<Chambre> findAllByType(String type) { return chambreRepository.findAllByType(type); }

    // trouver les réservations qui ne chevauchent pas les dates spécifiées
    2 usages
    public List<Chambre> getAllAvailableChambresByDates(LocalDate dateArrivee, LocalDate dateDepart) {...}
```

Ces services seront exposés par le contrôleur **ChambreController**

```
@CrossOrigin("*")
@RestController
@RequestMapping("/chambre")
public class ChambreController {

    @Autowired
    private ChambreService chambreService;

    @Autowired
    private ReservationService reservationService;

    @GetMapping("/{id}")
    public ResponseEntity<Chambre> getChambre(@PathVariable Integer id) {...}

    @GetMapping()
    public ResponseEntity<List<Chambre>> getAllChambres() {...}

    @GetMapping("/disponible")
    public ResponseEntity<List<Chambre>> getAllAvailableChambres() {...}

    @PostMapping
    public ResponseEntity<Chambre> createChambre(@RequestBody Chambre chambre) {...}

    @PutMapping("/{id}")
    public ResponseEntity<Chambre> updateChambre(@PathVariable Integer id, @RequestBody Chambre chambre) {...}

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteChambre(@PathVariable Integer id) {...}

    @GetMapping("/disponible/date")
    public ResponseEntity<List<Chambre>> getAllAvailableChambresParDate(
        @RequestParam @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate dateArrivee,
        @RequestParam @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate dateDepart) {...}

    @GetMapping("/type/{type}")
    public ResponseEntity<List<Chambre>> getAllAvailableChambresParType(@PathVariable String type) {...}

    @GetMapping("/disponible/type/{type}")
    public ResponseEntity<List<Chambre>> getAllAvailableChambresDisponiblesParType(@PathVariable String type) {...}

    @GetMapping("/disponible/type/{type}/date")
    public ResponseEntity<List<Chambre>> getAllAvailableChambresDisponiblesParDateParType
        (@PathVariable String type,
         @RequestParam @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate dateArrivee,
         @RequestParam @DateTimeFormat(pattern = "yyyy-MM-dd") LocalDate dateDepart
        ) {...}
```



## 4. Réservations

### reservation-controller

GET	/reservation	⌵
PUT	/reservation	⌵
POST	/reservation	⌵
GET	/reservation/{idReservation}	⌵
DELETE	/reservation/{idReservation}	⌵
GET	/reservation/guest/{username}	⌵

**GET /reservation** : pour accéder à toutes les réservations

**PUT /reservation** : pour modifier une réservation

**POST /reservation** : pour ajouter une réservation

**GET /reservation/{idReservation}** : pour accéder à une réservation par id

**DELETE /reservation/{idReservation}** : pour supprimer une réservation par id

**GET /reservation/guest/{username}** : pour supprimer une réservation par id

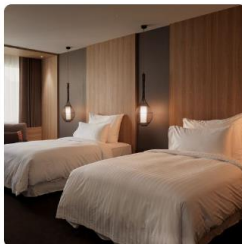
Un utilisateur peut accéder l'historique de ses réservations en utilisant l'API **GET /reservation/guest/{username}**

[Chambres](#)[Mes réservations](#)[Modifier profile](#)[Déconnexion](#)

## Historique des réservations

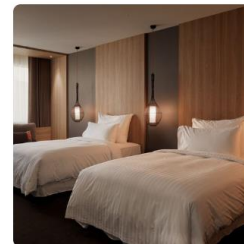
### Réservation n.37

Date de début: 2026-02-02  
Date de fin: 2027-03-03  
Chambre: Double  
Nombre de personnes: 1  
Prix: 304 €  
Status: ON\_HOLD



### Réservation n.40

Date de début: 2026-02-02  
Date de fin: 2027-03-03  
Chambre: Double  
Nombre de personnes: 1  
Prix: 304 €  
Status: CONFIRMED



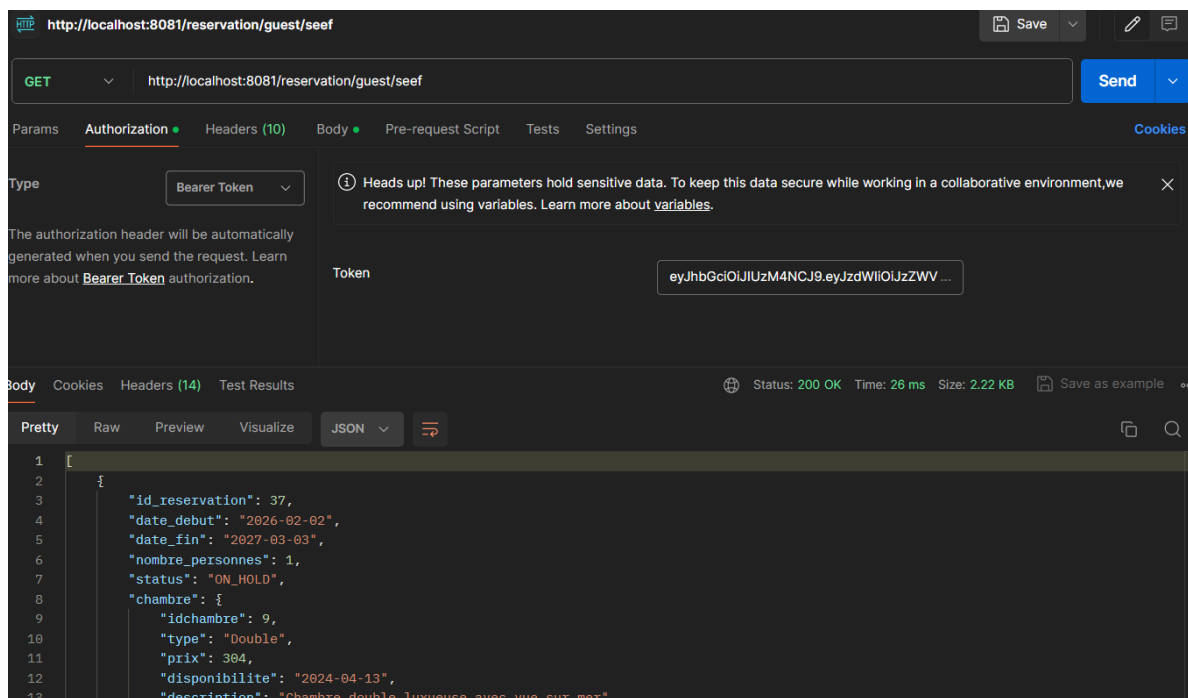
### Réservation n.45

Date de début: 2025-01-01  
Date de fin: 2029-01-03  
Chambre: Suite  
Nombre de personnes: 9  
Prix: 460 €  
Status: ON\_HOLD



Cette API est protégée : seulement l'utilisateur concerné et l'admin peuvent accéder ces réservations.

Test avec Postman :



Si on utilise un token autre que celui de l'utilisateur concerné et l'administrateur : **403 Forbidden**

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8081/reservation/guest/seef`
- Method:** GET
- Authorization:** Bearer Token
- Token:** `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ0IiwiaWF0IjoxNzE3MTE3OTUwLCJleHAiOiJlMTMxOTg5NTB9LmVudGppABpayoRk2ZLZVHb5aynG-xmuGW4Hxtxm8CrSBFoyNkFTgS3TKvDOWA0x6P`
- Status:** 403 Forbidden
- Time:** 18 ms
- Size:** 389 B

A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables."

Création d'une réservation :

Suite à la création d'une réservation le champ **disponibilite** de la chambre sera changé

The screenshot shows a web application interface for a hotel reservation. The background is a blurred image of a hotel interior. The reservation form is titled "Formulaire de Réservation" and contains the following fields:

- Date de début :** 01/01/2026 01:00 AM
- Date de fin :** 01/01/2027 01:00 AM
- Nombre de personnes :** 14
- Numéro de la chambre :** 19
- Type de chambre :** Suite
- Prix :** 460
- Nom d'utilisateur :** seef
- Email :** seef@seef.com

A blue "Réserver" button is located at the bottom of the form.

Avec Postman :

The screenshot shows a Postman interface for a POST request to `http://localhost:8081/reservation`. The request body is a JSON object:

```
{
  "date_debut": "2026-01-01T01:00",
  "date_fin": "2027-01-01T01:00",
  "nombre_personnes": "14",
  "status": "ON_HOLD",
  "chambre": {
    "idchambre": 19,
    "type": "Suite",
  }
}
```

The response status is `200 OK` with a time of `34 ms` and a size of `1.02 KB`. The response body is a JSON object:

```
{
  "id_reservation": 48,
  "date_debut": "2026-01-01",
  "date_fin": "2027-01-01",
  "nombre_personnes": 14,
  "status": "ON_HOLD",
}
```

Below the JSON response, the data is displayed in a table format:

48	2026-01-01	2027-01-01	19	9	14	ON_HOLD
----	------------	------------	----	---	----	---------

L'administrateur peut consulter toutes les réservations ainsi que les modifier ou les supprimer

The header of the application shows the user `seef` and navigation links: `Chambres`, `Mes réservations`, `Modifier profile`, and `Déconnexion`.

## Admin - Réservations

ID	User	Date de début	Date de fin	Chambre	Nb pers	Prix	STATUS	Actions	
37	seef	2026-02-02	2027-03-03	Double	1	304	ON_HOLD	Modifier	Supprimer
48	seef	2026-01-01	2027-01-01	Suite	14	460	ON_HOLD	Modifier	Supprimer
40	seef	2026-02-02	2027-03-03	Double	1	304	CONFIRMED	Modifier	Supprimer
43	admin	2026-01-01	2028-01-01	Double	1	304	ON_HOLD	Modifier	Supprimer
45	seef	2025-01-01	2029-01-03	Suite	9	460	ON_HOLD	Modifier	Supprimer

Les APIs de ces fonctionnalités sont protégées, seulement l'administrateur peut les accéder

Test avec Postman :

Avec le token d'admin,

### Accès de toutes les réservations

GET `http://localhost:8081/reservation` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Token `eyJhbGciOiJIUzI4NCJ9.eyJzdWiiOiJhZG1p...`

Body Cookies Headers (14) Test Results

Status: 200 OK Time: 37 ms Size: 3.43 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_reservation": 37,
3   "date_debut": "2026-02-02",
4   "date_fin": "2027-03-03",
5   "nombre_personnes": 1,
6   "status": "ON_HOLD",
7   "chambre": {
8     "idchambre": 9,
9     "type": "Double",
10    "prix": 304,
```

### Accès d'une reservation par id

HTTP `http://localhost:8081/reservation/37` Save

GET `http://localhost:8081/reservation/37` Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Type Bearer Token

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Token `eyJhbGciOiJIUzI4NCJ9.eyJzdWiiOiJhZG1p...`

Body Cookies Headers (14) Test Results

Status: 200 OK Time: 22 ms Size: 1.01 KB Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_reservation": 37,
3   "date_debut": "2026-02-02",
4   "date_fin": "2027-03-03",
5   "nombre_personnes": 1,
6   "status": "ON_HOLD",
7   "chambre": {
8     "idchambre": 9,
9     "type": "Double",
10    "prix": 304,
```

## Modification d'une reservation

PUT ▼ http://localhost:8081/reservation Send ▼

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "id_reservation": 37,
3   "date_debut": "2026-02-02",
4   "date_fin": "2027-03-03",
5   "nombre_personnes": 1,
6   "status": "CONFIRMED",
7   "chambre": {
8     "idchambre": 9,
9     "type": "Double",
10    "prix": 384,
11    "disponibilite": "2024-04-13",
12    "description": "Chambre double luxueuse avec vue sur mer",
13    "image": "https://i.pinimg.com/736x/93/4d/b7/934db7616c51a235d580c2e7fd589bdc.jpg"
14  }
15 }
```

Body Cookies Headers (14) Test Results 🌐 Status: 200 OK Time: 26 ms Size: 1.01 KB 💾 Save as example ⋮

Pretty Raw Preview Visualize **JSON** ▼ 🔍

```
1 {
2   "id_reservation": 37,
3   "date_debut": "2026-02-02",
4   "date_fin": "2027-03-03",
5   "nombre_personnes": 1,
6   "status": "CONFIRMED",
7   "chambre": {
8     "idchambre": 9,
9     "type": "Double",
10    "prix": 384,
11    "disponibilite": "2024-04-13",
12    "description": "Chambre double luxueuse avec vue sur mer",
13    "image": "https://i.pinimg.com/736x/93/4d/b7/934db7616c51a235d580c2e7fd589bdc.jpg"
14  }
15 }
```

id_reservation	date_debut	date_fin	idchambre	iduser	nombre_personnes	status
37	2026-02-02	2027-03-03	9	9	1	CONFIRMED

## Suppression d'une reservation

HTTP 🔗 http://localhost:8081/reservation 📁 Save ✎ 💬

DELETE ▼ http://localhost:8081/reservation Send ▼

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies

Query Params

	Key	Value	Description	⋮ Bulk Edit
	Key	Value	Description	

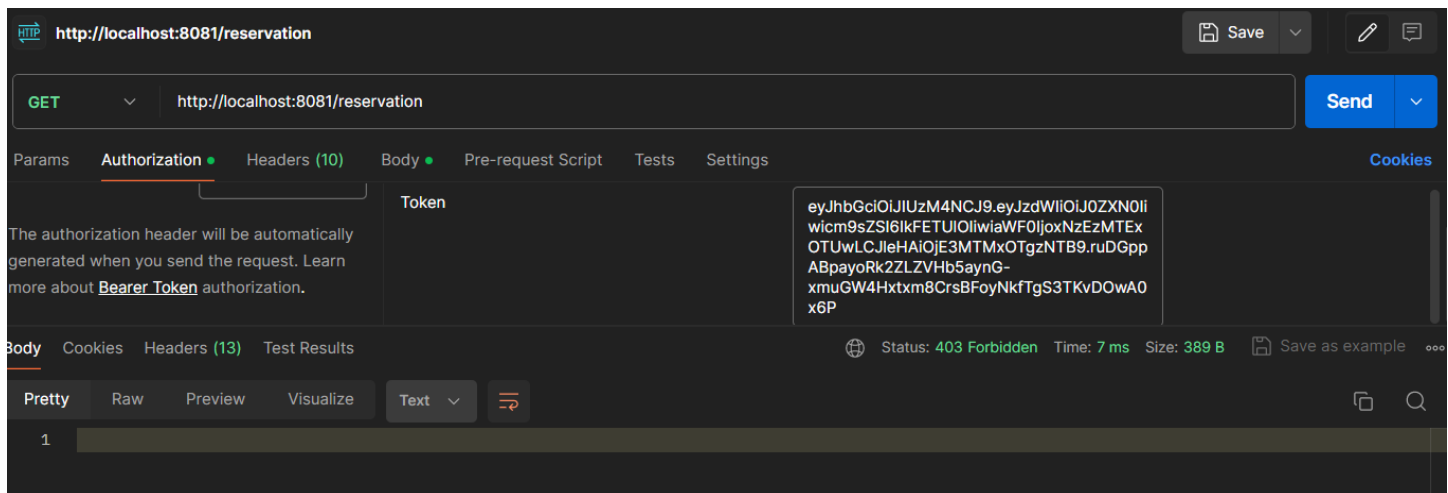
Body Cookies Headers (14) Test Results 🌐 Status: 403 Forbidden Time: 15 ms Size: 412 B 💾 Save as example ⋮

Pretty Raw Preview Visualize **Text** ▼ 🔍

```
1
```

Suite à la suppression d'une réservation, le champ **disponibilite** de la chambre concernée est changée à la date actuelle.

Si on accède à une API avec un token autre que celui de l'admin, **403 Forbidden** :



## Implémentation

Le modèle est **Reservation** dans le package **models**

Les services sont implémentés dans **ReservationService** qui utilise **ReservationRepository**

```
4 usages
public interface ReservationRepository extends JpaRepository<Reservation, Integer> {
    1 usage
    List<Reservation> findAllByUser(MyUser user);

    1 usage
    @Query("SELECT r FROM Reservation r " +
        "WHERE (:dateArrivee BETWEEN r.date_debut AND r.date_fin " +
        "OR :dateDepart BETWEEN r.date_debut AND r.date_fin " +
        "OR r.date_debut BETWEEN :dateArrivee AND :dateDepart " +
        "OR r.date_fin BETWEEN :dateArrivee AND :dateDepart)")
    List<Reservation> findOverlappingReservations(@Param("dateArrivee") LocalDate dateArrivee,
        @Param("dateDepart") LocalDate dateDepart);

    1 usage
    @Query("SELECT r FROM Reservation r " +
        "WHERE r.chambre.idchambre = :chambreId " +
        "AND ((r.date_debut BETWEEN :startDate AND :endDate) " +
        "OR (r.date_fin BETWEEN :startDate AND :endDate))")
    List<Reservation> findOverlappingReservationsParChambre(
        @Param("startDate") LocalDate startDate,
        @Param("endDate") LocalDate endDate,
        @Param("chambreId") Integer chambreId
    );

    2 usages
    void deleteAllByUser(MyUser user);

    1 usage
    void deleteAllByChambreIdchambre(Integer idchambre);
}
```

**findOverlappingReservations** et **findOverlappingReservationsParChambre** permettent de trouver les autres réservations qui chevauchent avec la date d'arrivée et de départ pour éviter de créer une réservation pour une chambre qui n'est pas disponible.

```
6 usages
@Service
public class ReservationService {

    @Autowired
    private ReservationRepository reservationRepository;

    @Autowired
    private ChambreRepository chambreRepository;

    @Autowired
    private MyUserRepository userRepository;

    1 usage
    public Reservation createReservation(Reservation reservation) {...}

    1 usage
    public Reservation getReservationById(Integer id) {
        return reservationRepository.findById(id).orElse( other: null);
    }

    1 usage
    public Reservation updateReservation(Reservation reservation) {
        return reservationRepository.save(reservation);
    }

    1 usage
    public void deleteReservation(Integer id) {...}

    1 usage
    public List<Reservation> getAllReservationsByUsername(String username) {...}

    no usages
    public void deleteAllReservationsByUsername(String username) {...}

    1 usage
    public List<Reservation> getAllReservations() { return reservationRepository.findAll(); }

    1 usage
    public void deleteAllByUser(MyUser user) { reservationRepository.deleteAllByUser(user); }

    1 usage
    public void deleteAllByChambreid(Integer chambreId) {...}
```



Les services sont exposés avec **ReservationController**

```
@CrossOrigin("*")
@RestController
@RequestMapping(⌚"/reservation")
public class ReservationController {

    @Autowired
    private ReservationService reservationService;

    @GetMapping(⌚)
    public ResponseEntity<List<Reservation>> getAllReservations() {...}

    @GetMapping(⌚"/quest/{username}")
    public ResponseEntity<List<Reservation>> getAllReservationsByUsername(@PathVariable String username) {...}

    ⚡ @GetMapping(⌚"/{idReservation}")
    public ResponseEntity<Reservation> getReservation(@PathVariable Integer idReservation) {...}

    @PostMapping(⌚)
    public ResponseEntity<Reservation> createReservation(@RequestBody Reservation reservation) {...}

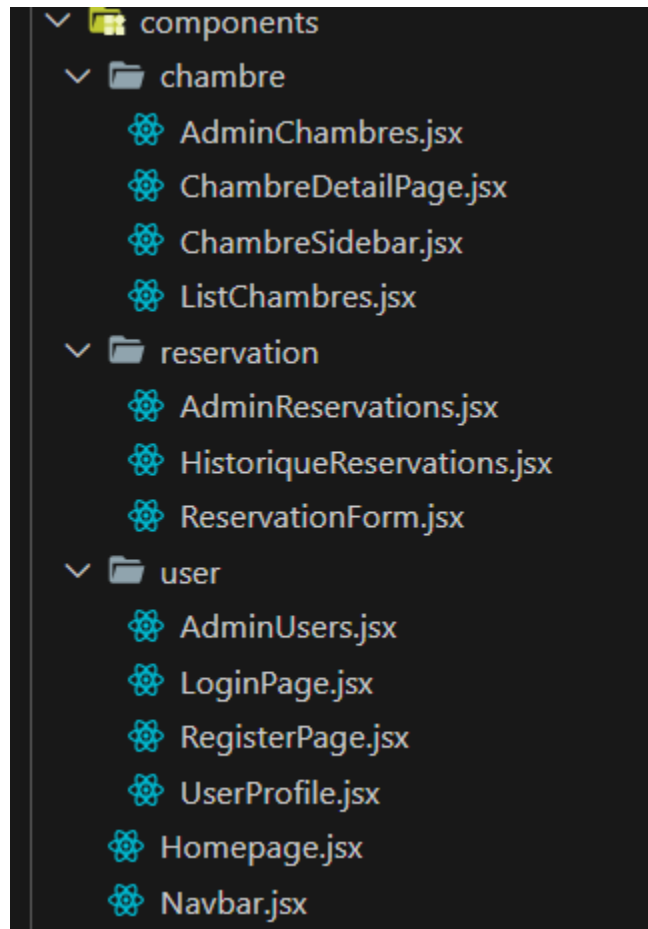
    @DeleteMapping(⌚"/{idReservation}")
    public ResponseEntity<Void> deleteReservation(@PathVariable Integer idReservation) {...}

    @PutMapping(⌚)
    public ResponseEntity<Reservation> updateReservation(@RequestBody Reservation reservation) {...}

    2 usages
```

## IV- Frontend

Le frontend est développé en utilisant React, Tailwindcss pour le styling et Axios pour les différents requêtes pour l'interaction avec le backend.



Le routing se fait avec le package **react-router-dom**

```
function App() {
  return (
    <>
      <ToastContainer />
      <Navbar />
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<HomePage />}></Route>
          <Route path="/chambre" element={<ListChambres />}></Route>
          <Route path="/chambre/:chambreId" element={<ChambreDetailPage />} />
          <Route path="/historique" element={<HistoriqueReservations />} />
          <Route
            path="/admin/reservations"
            element={<AdminReservations />}
          ></Route>
          <Route path="/admin/users" element={<AdminUsers />}></Route>
          <Route path="/admin/chambres" element={<AdminChambres />}></Route>
          <Route path="/reserver" element={<ReservationForm />}></Route>
          <Route path="/login" element={<LoginPage />} />
          <Route path="/register" element={<RegisterPage />} />
          <Route path="/profile" element={<UserProfile />} />
        </Routes>
      </BrowserRouter>
    </>
  );
};
```

Exemple d'envoi d'une requête avec Axios :

```
const handleSubmit = async (e) => {
  e.preventDefault();
  try {
    console.log("formdata:", formData);
    const response = await axios.post(
      "http://localhost:8081/reservation",
      formData,
      {
        headers: {
          Authorization: `Bearer ${token}`,
        },
      }
    );
    console.log("Reservation created:", response.data);
    toast.success("Réservation créée avec succès", {
      position: "bottom-center",
    });
  } catch (error) {
    console.error("Error creating reservation:", error);
    toast.error("Error creating reservation, check dates", {
      position: "bottom-center",
    });
  }
};
```