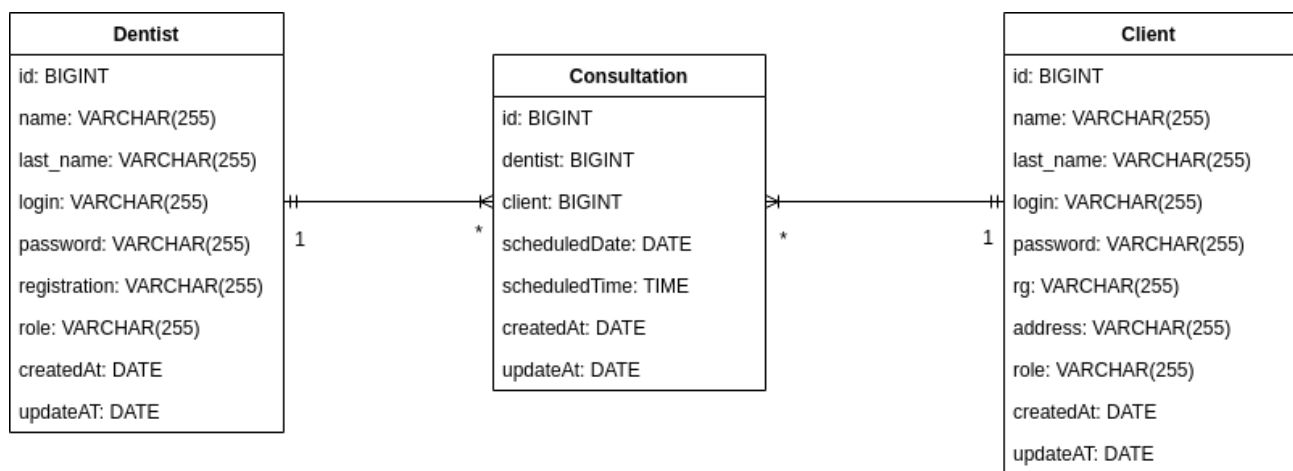


Grupo 09 – Projeto Integrador
Alunos:
Carlos Alberto Filho
Eduardo Ananias da silva
Denis Carbone
Salomão Lopes Dos Santos

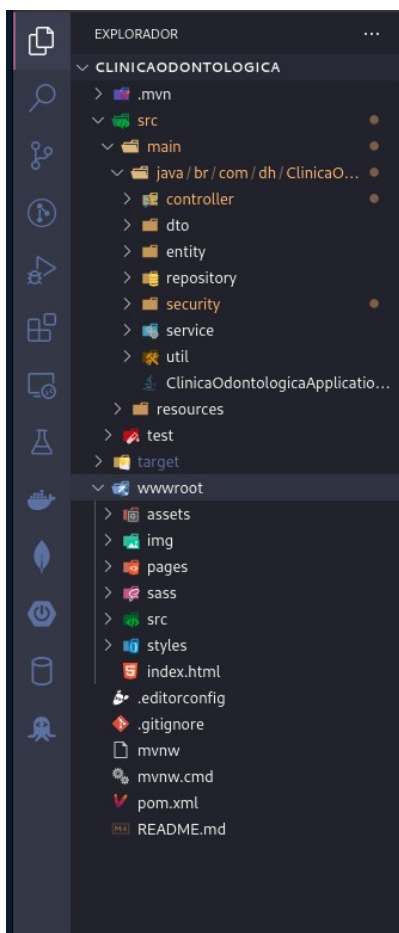
GitHub

O projeto está no link do GitHub - <https://github.com/carlosalbertofilho/ClinicaOdontologica>

Diagrama do Banco de Dados



Estrutura do projeto



- `src/main/java/br/com/dh/ClinicaOdontologica`

Código Java do Projeto está dividido em:

- **controller** – ControllerRest da Aplicação
 - **dto** – Objetos de mapeamento entre o banco de dados e as chamadas da API.
 - **entity** – Entidades de modelagem do Banco de Dados
 - **repository** – Interfaces do Spring JPA mapeadas com as Entidades do DB.
 - **security** – Classes do Spring Security
 - **service** – Implementação das Interfaces do JPA com a lógica da aplicação.
 - **util** – Classes de mapeamento entre entidades e DTO.
 - **test** – Teste unitário dos Services
- **wwwroot**
Código do Front-end feito em HTML5 + JS Vanilla + CSS

Endpoints da Aplicação

Usuários de Testes

Para testar a aplicação utilize um dos seguintes usuários:

- ADMIN

Usuário: teste123

Senha: 12345678

- Client

Usuário: eduardo.ananias@teste.com

Senha: teste@12345

Usuário: carlos.filho@teste.com

Senha: teste@123456

Usuário: luna@teste.com

Senha: teste@123456

- Dentist

Usuário: salomao.santos@teste.com

Senha: teste@123456

Usuário: lucas@teste.com

Senha: teste@123456

Usuário: denis.carbone@teste.com

Senha: teste@123456

H2 Console está disponível em <http://localhost:8080/h2>

← → ↻ localhost:8080/h2/login.jsp?js...

English Preferences Tools Help

Login

Saved Settings: Generic H2 (Embedded)

Setting Name: Generic H2 (Embedded) Save Remove

Driver Class: org.h2.Driver

JDBC URL: jdbc:h2:mem:ProjetoIntegradorDB

User Name: sa

Password:

Connect Test Connection

JDBC URL –

jdbc:h2:mem:ProjetoIntegradorDB

Auto complete
Max rows: 1000
 Auto complete Auto select On

jdbc:h2:mem:ProtegeReporterGRD

CLIENT

CONSULTATION

DENTIST

USER_ADMIN

INFORMATION_SCHEMA

Users

HD 21.234 (2022-09-12)

Run | Run Selected | Auto complete | Clear | SQL statement

SELECT * FROM CLIENT

SELECT * FROM CONSULTATION

SELECT * FROM DENTIST

SELECT * FROM USER_ADMIN

SELECT * FROM INFORMATION_SCHEMA

Users

HD 21.234 (2022-09-12)

SELECT * FROM CLIENT:

ID	ADDRESS	CREATED_AT	LAST_NAME	LOGIN	NAME	PASSWORD	REGISTRATION_ROLE	ROLE
1	Rua 10	2022-09-13	Aracelis	aracelis.aracelis@brite.com	Carolina	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE
2	Rua 10	2022-09-22	Filho	carlos.filho@brite.com	Carlos	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE
3	Rua 12	2022-09-22	Techica	luna@brite.com	Luna	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE
4	Rua 1	2022-09-26	Filho	carlos.filho@brite.com	Carlos	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE

(4 rows, 18 ms)

SELECT * FROM DENTIST:

ID	CREATED_AT	LAST_NAME	LOGIN	NAME	PASSWORD	REGISTRATION_ROLE	ROLE
1	2022-09-15	Santos	salmos.santos@brite.com	Salmos	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE
2	2022-09-26	Catharine	carlos.catharine@brite.com	Catharine	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE
3	2022-09-26	Guarules	lucas@brite.com	Lucas	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE
4	2022-09-26	meiro	luna.meiro@brite.com	Luna	\$2a\$10\$9Wp4d5Gf9P1Q4Z03M1hL6wCzUd8Z7T7e5uH9Qy5S8UW6wETC	123456789	ROLE

(4 rows, 12 ms)

SELECT * FROM CONSULTATION:

ID	CREATED_AT	SCHEDULED_DATE	SCHEDULED_TIME	UPDATE_AT	CLIENT_ID	DENTIST_ID
1	2022-09-26	2022-09-22	10:30:00	2022-09-26	2	2
2	2022-09-26	2022-09-26	12:00:00	2022-09-26	3	3
3	2022-09-26	2022-10-01	12:00:00	2022-09-26	3	3

Exemplo de Chamadas aos Endpoints

Admin user

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/login`. The request body is a JSON object: `{ "username": "teste123", "password": "12345678" }`. The response is a 200 OK status with a response time of 4273 ms. The response body is a JSON object: `{ "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJBcGkgRGggRWNVbW11", "type": "Bearer " }`.

POST `http://localhost:8080/login` Send

Description Headers Query Body Auth

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {
2   "username": "teste123",
3   "password": "12345678"
4 }
```

200 OK · 4273 ms

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJBcGkgRGggRWNVbW11
3   "type": "Bearer "
4 }
```

Filter expression...

JsonPath expressions e.g. \$.store.book[0].title

Request snippet TypeScript (Fetch) </> Copy

JSON to TypeScript </> Copy

Client User

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/login`. The request body is a JSON object: `{ "username": "eduardo.ananias@teste.com", "password": "teste@12345" }`. The response is a 200 OK status with a response time of 656 ms. The response body is a JSON object: `{ "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJBcGkgRGggRWNVbW11", "type": "Bearer " }`.

POST `http://localhost:8080/login` Send

Description Headers Query Body Auth

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {
2   "username": "eduardo.ananias@teste.com",
3   "password": "teste@12345"
4 }
```

200 OK · 656 ms

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJBcGkgRGggRWNVbW11
3   "type": "Bearer "
4 }
```

Filter expression...

JsonPath expressions e.g. \$.store.book[0].title

Request snippet TypeScript (Fetch) </> Copy

JSON to TypeScript </> Copy

Dentist User

The screenshot shows a Postman interface with a tab labeled 'login'. The request is a POST to 'http://localhost:8080/login'. The body is a JSON object with 'username' and 'password' fields. The response is a 200 OK status with a 'Bearer' token. The 'JSON Text' view is selected, showing the token and type.

POST http://localhost:8080/login Send

200 OK · 468 ms

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 {
2   "token": "eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJBcGkgRGggRWVnbW11
3   "type": "Bearer "
4 }
```

1 {
2 "username": "salomao.santos@teste.com",
3 "password": "teste@123456"
4 }

Request snippet TypeScript (Fetch) </> </>

JSON to TypeScript </> </>

Listar Cliente

The screenshot shows a Postman interface with a tab labeled 'Listar Clientes'. The request is a GET to 'http://localhost:8080/api/cliente'. The response is a 200 OK status with a list of client data. The 'JSON Text' view is selected, showing the list of clients.

GET http://localhost:8080/api/cliente Send

200 OK · 1156 ms

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 [
2   {
3     "id": 1,
4     "name": "Eduardo",
5     "lastName": "Ananias",
6     "login": "eduardo.ananias@teste.com",
7     "address": "Rua 50",
8     "role": "ROLE_CLIENT",
9     "createdAt": "2022-09-19",
10    "updatedAt": "2022-09-26"
11  },
12  {
13    "id": 2,
```

Token

eyJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJBcGkgRGggRWVnbW11cmNliiwic3

Default authentication scheme is set to "Bearer". Change this scheme name under "show more".

Show More

Criar Cliente

The screenshot shows the Postman interface for the 'Criar Clientes' endpoint. The method is POST and the URL is http://localhost:8080/api/cliente. The request body is in JSON format, containing fields: name, lastName, login, password, rg, and address. The response status is 201 Created with a 1033 ms response time. The response body is a JSON object with fields: id, name, lastName, login, address, role, createdAt, and updatedAt.

POST http://localhost:8080/api/cliente Send

Description Headers Query Body Auth

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Pretty Print JSON

```
1 {
2   "name": "carlos",
3   "lastName": "filho",
4   "login": "carlos.filho@teste.com",
5   "password": "123456",
6   "rg": "1234567890",
7   "address": "Rua 1"
8 }
```

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 {
2   "id": 4,
3   "name": "carlos",
4   "lastName": "filho",
5   "login": "carlos.filho@teste.com",
6   "address": "Rua 1",
7   "role": "ROLE_CLIENT",
8   "createdAt": "2022-09-26",
9   "updatedAt": null
10 }
```

Filter expression...

JsonPath expressions e.g. \$.store.book[0].title

Request snippet TypeScript (Fetch) </> Copy

JSON to TypeScript </> Copy

Listar Dentistas

The screenshot shows the Postman interface for the 'Listar Dentistas' endpoint. The method is GET and the URL is http://localhost:8080/api/dentista. The authentication is set to Bearer with a token. The response status is 200 OK with a 336 ms response time. The response body is a JSON array containing two objects, each with fields: id, name, lastName, login, role, createdAt, and updatedAt.

GET http://localhost:8080/api/dentista Send

Description Headers Query Body Auth

None Basic Bearer OAuth 1 OAuth 2

Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzcmMiOiJhcGgRGggRWNVbW1lcmNliiwic3

Default authentication scheme is set to "Bearer ". Change this scheme name under "show more".

Show More

Info Request Response

Headers Text JSON Tree JSON Text Raw

```
1 [
2   {
3     "id": 1,
4     "name": "Salomão",
5     "lastName": "Santos",
6     "login": "salomao.santos@teste.com",
7     "role": "ROLE_DENTIST",
8     "createdAt": "2022-03-15",
9     "updatedAt": "2022-09-26"
10  },
11  {
12    "id": 2,
13    "name": "Denis",
14  }
15 ]
```

Filter expression...

JsonPath expressions e.g. \$.store.book[0].title

Request snippet TypeScript (Fetch) </> Copy

JSON to TypeScript </> Copy

Criar Dentistas

The screenshot shows the Postman interface for a POST request to `http://localhost:8080/api/dentista`. The request body is a JSON object with the following fields: `name`, `lastName`, `login`, `password`, and `registration`. The response is a 201 Created status with a 938 ms response time. The response body is a JSON object with the following fields: `id`, `name`, `lastName`, `login`, `role`, `createdAt`, and `updatedAt`.

```
POST http://localhost:8080/api/dentista
```

Body (JSON):

```
{
  "name": "ruan",
  "lastName": "minto",
  "login": "ruan.minto@teste.com",
  "password": "123456",
  "registration": "9876554321"
}
```

Response (201 Created, 938 ms):

```
{
  "id": 4,
  "name": "ruan",
  "lastName": "minto",
  "login": "ruan.minto@teste.com",
  "role": "ROLE_DENTIST",
  "createdAt": "2022-09-26",
  "updatedAt": null
}
```

Listar Consultas

The screenshot shows the Postman interface for a GET request to `http://localhost:8080/api/consulta`. The request is authenticated using Bearer authentication with a token. The response is a 200 OK status with a 283 ms response time. The response body is a JSON array of two consultation objects, each with fields: `id`, `dentistID`, `clientID`, `createdAt`, `updatedAt`, `scheduledDate`, and `scheduledTime`.

```
GET http://localhost:8080/api/consulta
```

Auth (Bearer):

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXLTJ0IiwiaWF0IjoiMTY2MjY1MjY1In0=
```

Response (200 OK, 283 ms):

```
[
  {
    "id": 1,
    "dentistID": 2,
    "clientID": 2,
    "createdAt": "2022-09-26",
    "updatedAt": "2022-09-26",
    "scheduledDate": "2022-09-22",
    "scheduledTime": "10:30:00"
  },
  {
    "id": 2,
    "dentistID": 3,
    "clientID": 3,
    "createdAt": "2022-09-26",
    "updatedAt": "2022-09-26",
    "scheduledDate": "2022-09-22",
    "scheduledTime": "15:30:00"
  }
]
```

Criar Consultas

The screenshot shows a REST client interface with a dark theme. At the top, there are tabs for 'AddDataOnDB.java', 'login', 'Criar Consulta' (active), and 'Lista Consultas'. The main interface is divided into two main sections: the left for request configuration and the right for response details.

Request Configuration (Left):

- Method:** POST
- URL:** http://localhost:8080/api/consulta
- Body:** JSON (selected), with a 'Pretty Print JSON' button.
- Request Body (JSON):**

```
1 {
2   "dentistID": 2,
3   "clientID": 1,
4   "scheduledDate": "2022-10-01",
5   "scheduledTime": "12:20:00"
6 }
```

Response Details (Right):

- Status:** 201 Created, 524 ms
- Response Body (JSON Text):**

```
1 {
2   "id": 3,
3   "dentistID": 2,
4   "clientID": 1,
5   "createdAt": "2022-09-26",
6   "updatedAt": null,
7   "scheduledDate": "2022-10-01",
8   "scheduledTime": "12:20:00"
9 }
```

Bottom Bar:

- Request snippet:** TypeScript (Fetch)
- JSON to:** TypeScript