



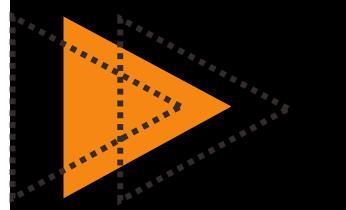
JAVA & JPA

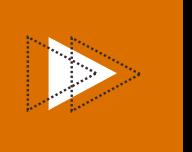
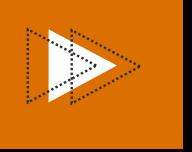
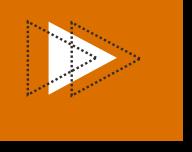
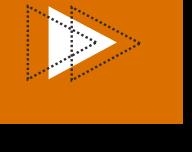
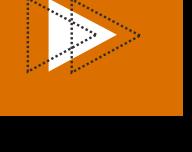


SISTEMA DE VIAGENS

Alunos: Jennifer Diehl, Matheus Porto e
Paulo Lanius

AGENDA



-  JPA
-  MYSQL
-  TEMA
-  DIAGRAMA
-  IMPLEMENTAÇÃO
-  DEMO



INTRODUÇÃO

Introdução

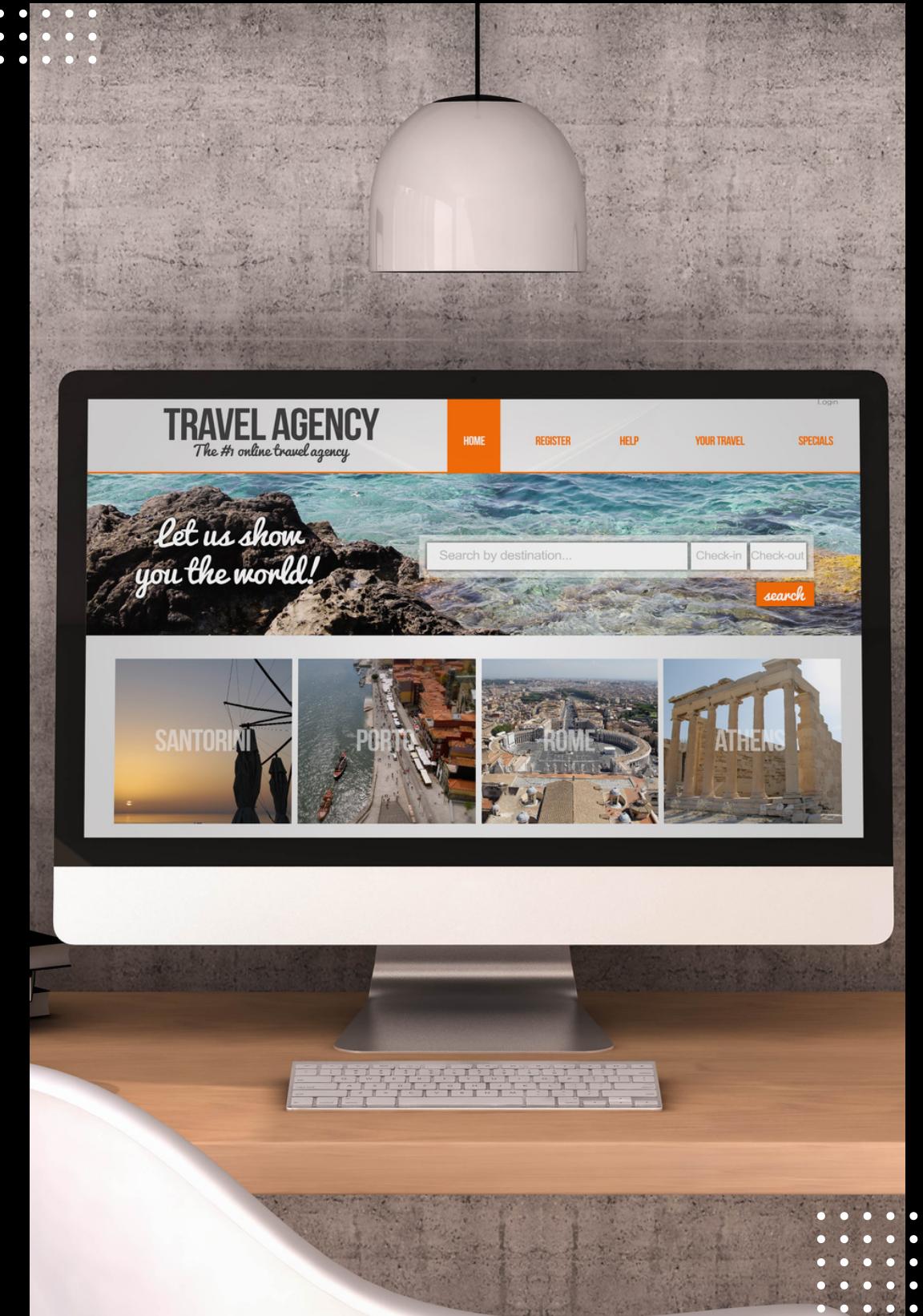
- JPA = Java Persistence API
- Framework leve para persistir objetos Java
- Amplamente utilizado
- Não é apenas um ORM, oferece diversas funcionalidade





HISTÓRICO

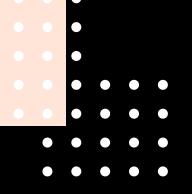
- Especificação Java EE 5
- Componentes como POJOS (Plain Java Objects)
- Surgiu da necessidade dos profissionais e das soluções para resolver problemas com persistência
- JPA 2.0

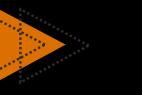




CARACTERÍSTICAS

- POJOS persistentes
- Consultas de objetos
- Configurações simples
- Integração e teste





BENEFÍCIOS

- Consultas com JPQL
- Abstração de banco de dados
- Produtividade
- Portabilidade

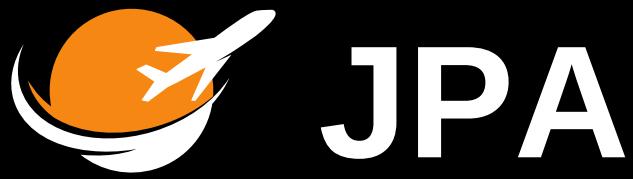




DESAFIOS

- Complexidade configuração e anotações para iniciantes
- Problemas de desempenho se não usado corretamente
- Curva de aprendizado
- Ferramentas e ecossistema





ANOTAÇÕES

@Entity, @Table, @Id, @GeneratedValue,
@Column, @OneToMany, @ManyToOne,
@JoinColumn, @Transient, @NamedQuery,
@Version, @Temporal, @Lob

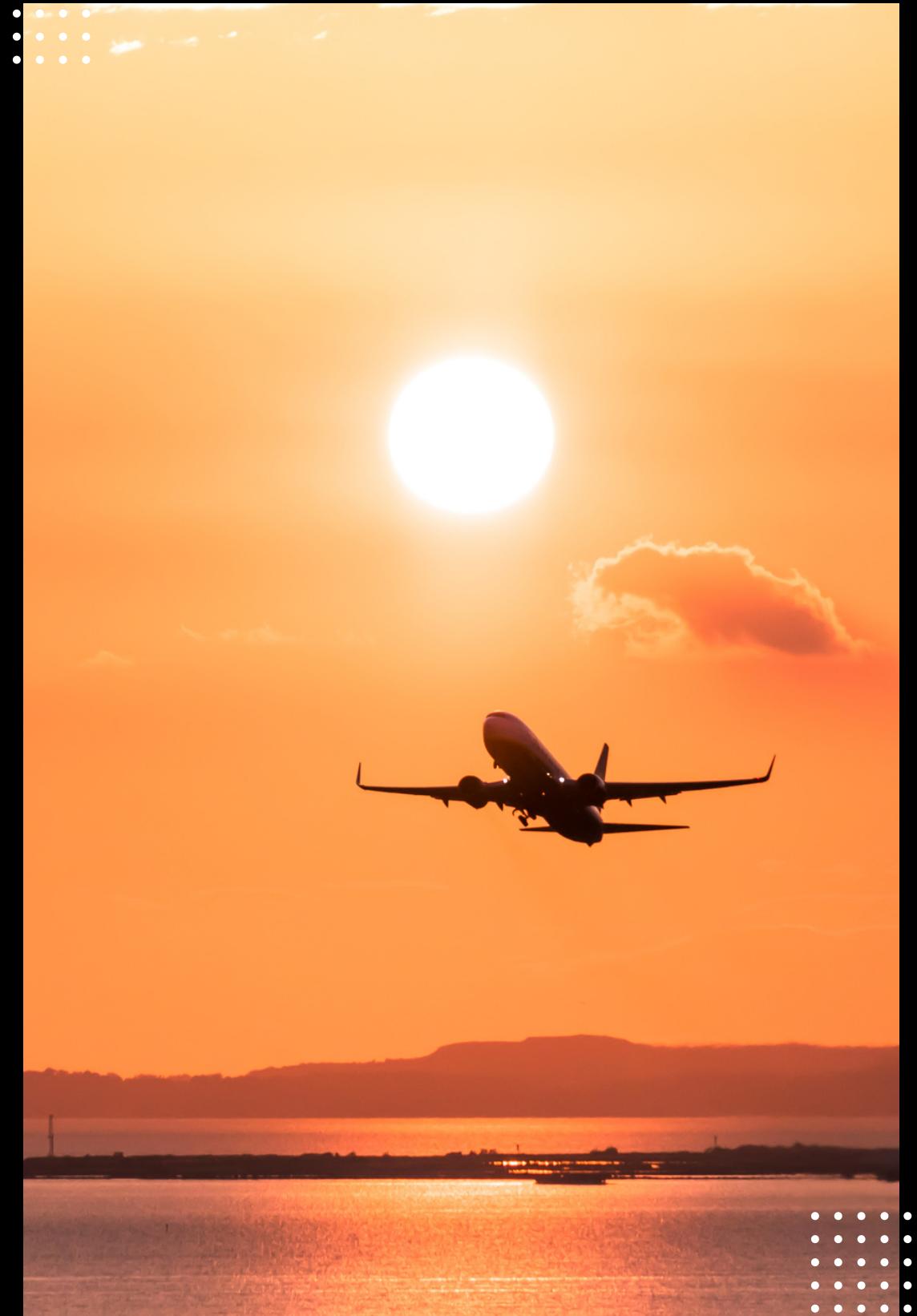
COMANDOS

EntityManager: persist, merge, remove, find,
createQuery, getTransaction, clear, refresh

Entity Transaction: begin, commit, rollback



- SGBD de código aberto
- Pertence a Oracle Corporation
- Amplamente utilizado e um dos mais populares
- Utiliza SQL como interface
- Tem suporte a várias plataformas
- Boa performance e baixa latência
- Comunidade ativa e vasta documentação
- Usado em aplicações web dinâmicas, sistemas de gerência de conteúdo, armazenamento de dados empresariais, aplicações móveis e outros.



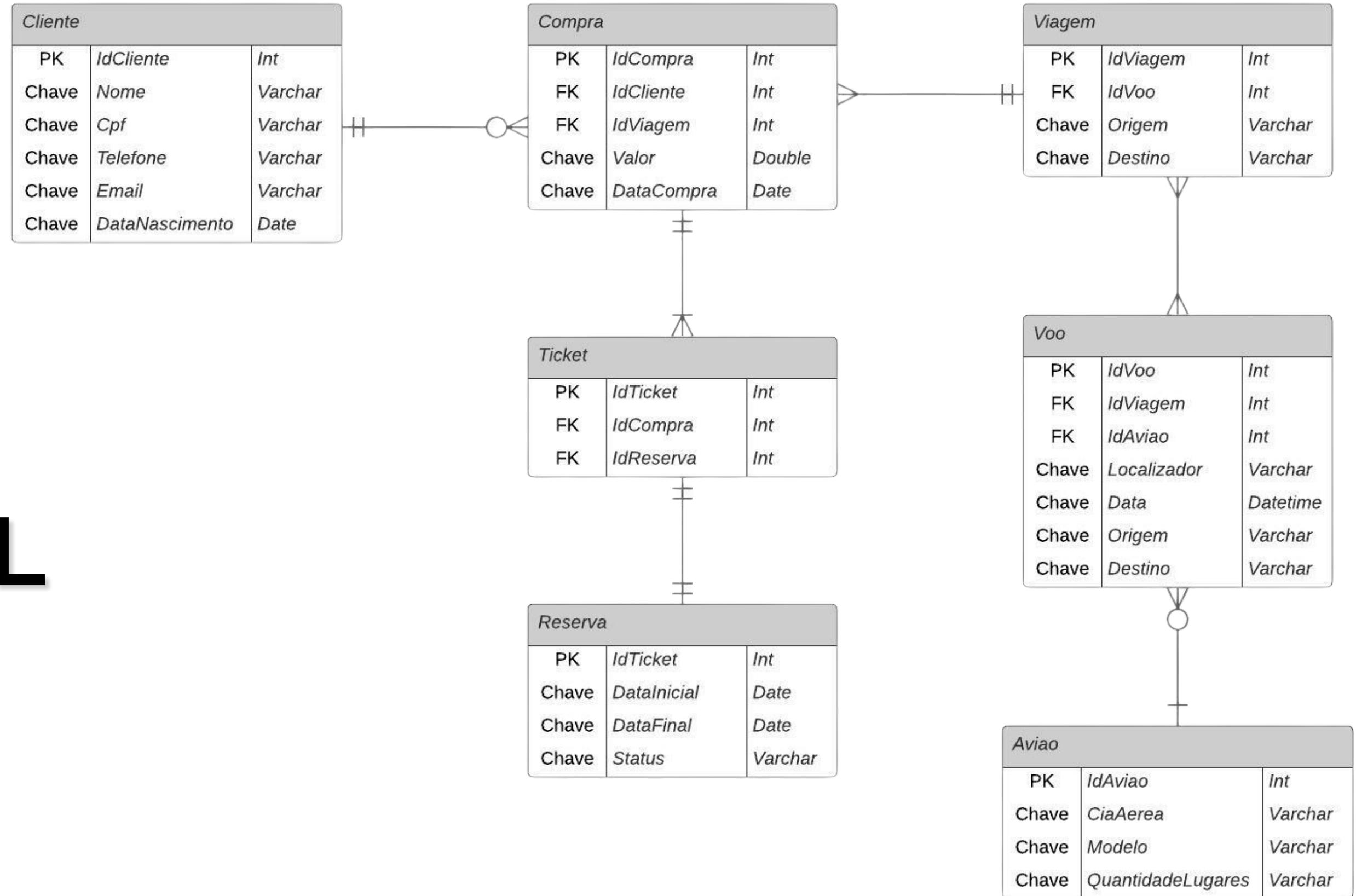


- Registrar novos usuários
- Realizar compras de pacotes
- Obter histórico de compras
- Obter informações do voo
- Realizar reservas





DIAGRAMA RELACIONAL





IMPLEMENTAÇÃO

The screenshot shows a Java code editor with three tabs open, each displaying a controller class from a Spring Boot application:

- ClienteController.java**: Handles client-related requests. It injects a `ClienteRepository` via constructor and uses it to save new clients and find one by ID.
- CompraController.java**: Handles purchase requests. It injects a `CompraRepository` and a `TicketRepository`. It creates a new purchase, associates it with tickets, and saves it to the database.
- ViagemController.java**: Handles travel requests. It injects a `AviaoRepository` and a `VooRepository`. It creates a new travel record, associates it with an aircraft, and saves it to the database.

The code editor also shows the project structure on the left, including packages for `java`, `resources`, and `static`, along with various repository interfaces and the main application class (`Homework2Application`).

```
java
com.db2.homework2
    controller
        ClienteController
        CompraController
        ViagemController
    models
        Aviao
        Cliente
        Compra
        Reserva
        Ticket
        Viagem
        Voo
    repository
        AviaoRepository
        ClienteRepository
        CompraRepository
        ReservaRepository
        TicketRepository
        ViagemRepository
        VooRepository
    Homework2Application
resources
    static
    templates
    application.properties
```

```
© ClienteController.java ×
© CompraController.java ×
© ViagemController.java ×
```

```
import com.db2.homework2.repository.ClienteRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@.RestController
@RequestMapping("/cliente")
public class ClienteController {

    private ClienteRepository repository;

    @PostMapping
    public ResponseEntity<String> cadastrarCliente(Cliente cliente) {
        repository.save(cliente);
        return ResponseEntity.ok("Cliente cadastrado com sucesso!");
    }

    @GetMapping("/{id}")
    public Cliente buscarCliente(@PathVariable Long id) {
        return repository.findById(id).orElse(null);
    }
}
```

```
no usages
@RestController
@RequestMapping("/compra")
public class CompraController {

    private CompraRepository compraRepository;
    private TicketRepository ticketRepository;
    private ReservaRepository reservaRepository;

    @PostMapping
    public ResponseEntity<Compra> registrarCompra(Compra novaCompra) {
        Compra novaCompra = compraRepository.save(novaCompra);
        List<Ticket> tickets = gerarTickets(novaCompra);
        for (Ticket ticket : tickets) {
            ticketRepository.save(ticket);
        }
        novaCompra.setTickets(tickets);
        novaCompra = compraRepository.save(novaCompra);
        return new ResponseEntity<>(novaCompra);
    }
}
```

```
no usages
@RestController
@RequestMapping("/viagem")
public class ViagemController {

    private AviaoRepository aviaoRepository;
    private VooRepository vooRepository;

    @PostMapping("/cadastrarViagem")
    public ResponseEntity<String> cadastrarViagem(Viagem viagem) {
        viagemRepository.save(viagem);
        return ResponseEntity.ok("Viagem cadastrada com sucesso!");
    }
}
```



IMPLEMENTAÇÃO

The screenshot shows a Java code editor with seven files open in tabs, each representing a class from a travel application. The files are:

- Aviao.java**: An entity class with a many-to-many relationship to **Voo**.
- Cliente.java**: An entity class.
- Compra.java**: An entity class.
- Reserva.java**: An entity class with a many-to-one relationship to **Ticket** and a one-to-many relationship to **Viagem**. It has an annotation **@JsonIgnore** on its **id** field.
- Ticket.java**: An entity class.
- Viagem.java**: An entity class with a many-to-many relationship to **Voo**.
- Voo.java**: An entity class.

The code uses various annotations from the **jakarta.persistence** and **jakarta.json** packages. The **Reserva.java** file is currently being edited, as indicated by the cursor and the highlighted **@JsonIgnore** annotation.

```
java
com.db2.homework2
    controller
        ClienteController
        CompraController
        ViagemController
    models
        Aviao
        Cliente
        Compra
        Reserva
        Ticket
        Viagem
        Voo
    repository
        AviaoRepository
        ClienteRepository
        CompraRepository
        ReservaRepository
        TicketRepository
        ViagemRepository
        VooRepository
    Homework2Application
resources
    static
    templates
    application.properties
```

File	Content Summary
Aviao.java	Entity class with many-to-many relationship to Voo.
Cliente.java	Entity class.
Compra.java	Entity class.
Reserva.java	Entity class with relationships to Ticket and Viagem, and an annotation @JsonIgnore on its id field.
Ticket.java	Entity class.
Viagem.java	Entity class with many-to-many relationship to Voo.
Voo.java	Entity class.



IMPLEMENTAÇÃO

```
✓ └─ java
    └─ com.db2.homework2
        └─ controller
            └─ ClienteController
            └─ CompraController
            └─ ViagemController
        └─ models
            └─ Aviao
            └─ Cliente
            └─ Compra
            └─ Reserva
            └─ Ticket
            └─ Viagem
            └─ Voo
        └─ repository
            └─ AviaoRepository
            └─ ClienteRepository
            └─ CompraRepository
            └─ ReservaRepository
            └─ TicketRepository
            └─ ViagemRepository
            └─ VooRepository
        └─ Homework2Application
    └─ resources
        └─ static
        └─ templates
        └─ application.properties
```

The screenshot shows a code editor with seven tabs open, each displaying a Java interface for a repository. The tabs are: AviaoRepository.java, ClienteRepository.java, CompraRepository.java, ReservaRepository.java, TicketRepository.java, ViagemRepository.java, and VooRepository.java. Each tab contains the following code:

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface AviaoRepository extends JpaRepository<Aviao, Long> {
9 }
10
```

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface ClienteRepository extends JpaRepository<Cliente, Long> {
9 }
10
```

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface CompraRepository extends JpaRepository<Compra, Long> {
9 }
10
```

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface ReservaRepository extends JpaRepository<Reserva, Long> {
9 }
10
```

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface TicketRepository extends JpaRepository<Ticket, Long> {
9 }
10
```

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface ViagemRepository extends JpaRepository<Viagem, Long> {
9 }
10
```

```
1 package com.db2.homework2.repository;
2
3 import com.db2.homework2.models.*;
4 import org.springframework.data.jpa.repository.JpaRepository;
5 import org.springframework.stereotype.Repository;
6
7 @Repository
8 public interface VooRepository extends JpaRepository<Voo, Long> {
9 }
10
```



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and 'TGA'. The main area shows a POST request titled 'Cadastrar cliente' with the URL 'http://localhost:8080/cliente'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2     "nome": "Paulo Lanius",  
3     "cpf": "772.638.860-90",  
4     "email": "paulo@example.com",  
5     "telefone": "+55123456789",  
6     "dataNascimento": "1998-02-11"  
7 }  
8
```

The response at the bottom indicates a 200 OK status with 74 ms and 195 B. The response body is: 'Cliente cadastrado com sucesso!'



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and 'TGA', and various API endpoints such as 'POST Cadastrar cliente', 'POST Cadastrar viagem', 'POST Cadastrar voo', etc. The main workspace shows a POST request to 'http://localhost:8080/viagem/cadastrar'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "voos": [  
3     {  
4       "idVoo": 6,  
5       "localizador": "ABC910"  
6     }  
]
```

The response status is 200 OK with 89 ms latency and 194 B size. The response body is: 'Viagem cadastrada com sucesso!'



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and 'TGA', and various endpoints such as 'POST Cadastrar cliente', 'POST Cadastrar viagem', 'POST Cadastrar voo', 'POST Cadastrar aviao', 'POST Registrar compra', 'POST Reservar Ticket', and 'GET Buscar cliente'. The main panel shows a POST request to 'http://localhost:8080/viagem/voo'. The 'Body' tab is selected, showing a JSON payload:

```
1  {
2    "idAviao": {
3      "idAviao": 6,
4    },
5    "localizador": "ABC1012",
6    "data": "2023-09-30T10:00:00",
7    "origem": "BARCELONA",
8    "destino": "POA"
9 }
```

The response status is 200 OK, with a response body: 'Voo cadastrado com sucesso!'



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and 'TGA'. The main area shows a POST request to 'http://localhost:8080/viagem/aviao'. The request body is set to 'JSON' and contains the following JSON:

```
1 {  
2     "ciaAerea": "Azul Airlines",  
3     "modelo": "Boeing 7223",  
4     "quantidadeLugares": 150  
5 }  
6
```

The response status is 200 OK with a time of 67 ms and a size of 194 B. The response body is: 'Avião cadastrado com sucesso!'



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and 'TGA', and environments. The main area shows a POST request to 'http://localhost:8080/compra'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "cliente": {  
3     "idCliente": 3  
4   },  
5   "viagem": {  
6     "idViagem": 3  
7   },  
8   "tickets": [  
9     {  
10        "idTicket": 3  
11      },  
12      {  
13        "idTicket": 4  
14      }  
15   ],  
16   "valor": 5800.00,  
17 }  
18  
19 {  
20   "idCompra": 6,  
21 }  
22 }
```

The response status is 201 Created with 81 ms and 408 B. Below the body, there are tabs for Pretty, Raw, Preview, Visualize, and JSON.



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and environments like 'TGA'. The main workspace shows a POST request for 'Reservar Ticket' with the URL `http://localhost:8080/compra/ticket/2`. The 'Body' tab is selected, showing JSON input:

```
1 {  
2   "dataInicial": "2023-09-25",  
3   "dataFinal": "2023-09-30",  
4   "status": "Reservado"  
5 }  
6
```

The bottom navigation bar includes links for 'Online', 'Find and replace', 'Console', 'Runner', 'Capture requests', 'Cookies', 'Trash', and a help icon.



ENDPOINTS

The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, API Network, Explore, a search bar, and various settings icons. The main workspace is titled "My Workspace". On the left sidebar, there are sections for Collections, Environments, and History. Under Collections, "Database II" is expanded, showing a folder named "TGA" which contains several POST requests: "Cadastrar cliente", "Cadastrar viagem", "Cadastrar voo", "Cadastrar aviao", "Registrar compra", "Reservar Ticket", and "Buscar cliente". The "Buscar cliente" endpoint is currently selected. The main panel displays an HTTP request configuration for a GET method to "http://localhost:8080/cliente/1". The "Headers" tab is active, showing "(6)" headers. The "Body" tab indicates "none". A note states "This request does not have a body". At the bottom, the response status is shown as 200 OK with 142 ms and 123 B. There are also links for "Save as Example", "Runner", "Capture requests", "Cookies", "Trash", and a help icon.



ENDPOINTS

The screenshot shows the Postman application interface. The left sidebar displays 'My Workspace' with collections like 'Database II' and environments like 'TGA'. The main workspace shows a POST request for 'Reservar Ticket' with the URL `http://localhost:8080/compra/ticket/2`. The 'Body' tab is selected, showing JSON data:

```
1 {  
2   "dataInicial": "2023-09-25",  
3   "dataFinal": "2023-09-30",  
4   "status": "Reservado"  
5 }  
6
```

The bottom navigation bar includes links for 'Online', 'Find and replace', 'Console', 'Runner', 'Capture requests', 'Cookies', 'Trash', and a help icon.



SCHEMAS

CLIENTE

10 • `SELECT * FROM ARCHIVE.cliente;`

11 • `SELECT * FROM ARCHIVE.cliente;`

id_cliente	cpf	data_nascimento	email	nome	telefone
1	299.432.990-53	1999-06-22	jennifer@example.com	Jennifer Diehl	+55123456789
2	467.105.810-39	1999-05-14	matheus@example.com	Matheus Moraes Porto	+55123456789
3	772.638.860-90	1998-02-10	paulo@example.com	Paulo Lanius	+55123456789

TICKET

12 • `SELECT * FROM ARCHIVE.ticket;`

13 • `SELECT * FROM ARCHIVE.ticket;`

id_ticket	id_compra
1	3
2	4
3	5
4	6

RESERVA

13 • `SELECT * FROM ARCHIVE.reserva;`

14 • `SELECT * FROM ARCHIVE.reserva;`

id_ticket	data_final	data_inicial	status
4	2023-09-29	2023-09-24	Reservado

COMPRA

11 • `SELECT * FROM ARCHIVE.compra;`

12 • `SELECT * FROM ARCHIVE.compra;`

id_compra	data_compra	valor	id_cliente	id_viagem
3	2023-09-17	6000.00	1	1
4	2023-09-17	7000.00	2	2
5	2023-09-17	7000.00	2	2
6	2023-09-17	5800.00	3	3



SCHEMAS

AVIÃO

4 • `SELECT * FROM ARCHIVE.aviao;`

5 •

100% ⏪ 1:3

Result Grid Filter Rows: Search Edit:

id_aviao	cia_aerea	modelo	quantidade_lugares
1	Copa Airlines	Boeing 737	150
2	Copa Airlines	Boeing 768	150
3	American Airlines	Boeing 498	150
4	GOL Airlines	Boeing 733	150
5	Azul Airlines	Boeing 7323	150
6	Azul Airlines	Boeing 7223	150

VOO

9 • `SELECT * FROM ARCHIVE.voo;`

10 •

100% ⏪ 30:5

Result Grid Filter Rows: Search Edit: Export/Import:

id_voo	data	destino	localizador	origem	id_aviao
1	2023-09-18 07:00:00.000000	CANADA	ABC123	POA	1
2	2023-09-25 07:00:00.000000	POA	ABC456	CANADA	2
4	2023-09-25 07:00:00.000000	PARIS	ABC678	POA	4
5	2023-09-30 07:00:00.000000	POA	ABC789	PARIS	4
6	2023-09-30 07:00:00.000000	BARCELONA	ABC1011	POA	5
7	2023-09-30 07:00:00.000000	POA	ABC1012	BARCELONA	6

VIAGEM

7 • `SELECT * FROM ARCHIVE.viagem;`

8 •

100% ⏪ 1:6

Result Grid Filter Rows: Search

id_viagem	destino	origem
1	CANADA-POA	POA-CANADA
2	PARIS-POA	POA-PARIS
3	BARCELONA-POA	POA-BARCELONA

VIAGEM_VOO

7 • `SELECT * FROM ARCHIVE.viagem_voo;`

8 •

100% ⏪ 1:8

Result Grid Filter Rows: Search Ex

viagem_id	voo_id
1	1
1	2
2	4
2	5
3	6
3	7



CONSULTAS

CLIENTES E SUAS RESPECTIVAS COMPRAS

```
16    -- CUSTOMER/PURCHASE CONSULT
17 •  SELECT c.*, co.id_compra, co.valor AS valorCompra, co.data_compra, t.id_ticket
18   FROM ARCHIVE.cliente c
19   LEFT JOIN ARCHIVE.compra co ON c.id_cliente = co.id_cliente
20   LEFT JOIN ARCHIVE.ticket t ON co.id_compra = t.id_compra;
```

100% 1:28

Result Grid Filter Rows: Search Export:

id_cliente	cpf	data_nascimento	email	nome	telefone	id_compra	valorCompra	data_compra	id_ticket
1	299.432.990-53	1999-06-22	jennifer@example.com	Jennifer Diehl	+55123456789	3	6000.00	2023-09-17	1
2	467.105.810-39	1999-05-14	matheus@example.com	Matheus Moraes Porto	+55123456789	5	7000.00	2023-09-17	3
2	467.105.810-39	1999-05-14	matheus@example.com	Matheus Moraes Porto	+55123456789	4	7000.00	2023-09-17	2
3	772.638.860-90	1998-02-10	paulo@example.com	Paulo Lanius	+55123456789	6	5800.00	2023-09-17	4

CLIENTE ESPECÍFICO E SUAS COMPRAS

```
22    -- SPECIFIC CUSTOMER/PURCHASE CONSULT
23 •  SELECT c.*, co.id_compra, co.valor AS valorCompra, co.data_compra, t.id_ticket
24   FROM ARCHIVE.cliente c
25   LEFT JOIN ARCHIVE.compra co ON c.id_cliente = co.id_cliente
26   LEFT JOIN ARCHIVE.ticket t ON co.id_compra = t.id_compra
27   WHERE c.nome = 'Jennifer Diehl';
```

100% 1:21

Result Grid Filter Rows: Search Export:

id_cliente	cpf	data_nascimento	email	nome	telefone	id_compra	valorCompra	data_compra	id_ticket
1	299.432.990-53	1999-06-22	jennifer@example.com	Jennifer Diehl	+55123456789	3	6000.00	2023-09-17	1



CONSULTAS

CLIENTE COM TICKET RESERVADO

```
30 •  SELECT t.*, cl.nome, cl.cpf
31   FROM ARCHIVE.ticket t
32   INNER JOIN ARCHIVE.reserva r ON t.id_ticket = r.id_ticket
33   INNER JOIN ARCHIVE.compra c ON t.id_compra = c.id_compra
34   INNER JOIN ARCHIVE.cliente cl ON c.id_cliente = cl.id_cliente
35   WHERE cl.nome = 'Paulo Lanius' AND r.status = 'Reservado';
36
```

00% 33:27

Result Grid Filter Rows: Search Export:

id_ticket	id_compra	nome	cpf
4	6	Paulo Lanius	772.638.860-90

QUANTIDADE DE COMPRAS DE UM CLIENTE

```
38 •  SELECT COUNT(*) AS total_compras, c.nome
39   FROM ARCHIVE.cliente c
40   INNER JOIN ARCHIVE.compra co ON c.id_cliente = co.id_cliente
41   WHERE c.nome = 'Matheus Moraes Porto';
```

00% 59:35

Result Grid Filter Rows: Search Export:

total_compras	nome
2	Matheus Moraes Porto

VOOS PARA UMA VIAGEM DE POA-PARIS

```
44 •  SELECT v.*
45   FROM ARCHIVE.viagem_voo vv
46   JOIN ARCHIVE.viagem vi ON vv.viagem_id = vi.id_viagem
47   JOIN ARCHIVE.voo v ON vv.voo_id = v.id_voo
48   WHERE vi.origem = 'POA-PARIS' AND vi.destino = 'PARIS-POA';
49
```

100% 23:39

Result Grid Filter Rows: Search Export:

id_voo	data	destino	localizador	origem	id_aviao
4	2023-09-25 07:00:00.000000	PARIS	ABC678	POA	4
5	2023-09-30 07:00:00.000000	POA	ABC789	PARIS	4



OBRIGADO!



<https://github.com/jenniferdiehl/travel-agency-dbms>