



Universidade Federal do Sul e Sudeste do Pará
Faculdade de Computação e Engenharia Elétrica
Curso de Engenharia da Computação
Iago Costa das Flores - 201840601017
Juliana Batista da Silva - 201740601024

Sistemas Distribuídos
Trabalho avaliativo 02

Marabá
2021



Iago Costa das Flores - 201840601017
Juliana Batista da Silva - 201740601024

Sistemas Distribuídos
Trabalho Avaliativo 02

Relatório apresentado no curso de Engenharia da Computação, turma de 2018 como obtenção de nota parcial na disciplina de Sistemas Distribuídos, ministrada pelo Professor Dr. Warley Muricy Valente Junior.



Sumário

1 - Introdução	4
2 - Desenvolvimento do projeto	4
2.1 - Cliente	4
2.2 – Servidor	11
3 - Conclusão	16
4 - Referências	17

1 - Introdução

Este relatório tem como objetivo apresentar a solução encontrada pela equipe para o trabalho final 02 da disciplina de sistemas distribuídos. Desenvolver um sistema cliente-servidor para uma agência de turismo.

Requisitos da aplicação:

- Utilizar Web Services (SOAP ou REST) para prover a comunicação entre os clientes e o servidor da agência de turismo.
- Cliente e servidor devem estar implementados em linguagens distintas.

Métodos disponíveis no Servidor:

- Consulta e compra de passagens aéreas (fornecendo opção “somente ida” ou “ida e volta”, origem, destino, data da ida e data da volta, número e idade das pessoas e dados do cartão juntamente com a opção de parcelamento (se for efetuar a compra)).
- Consulta e compra de hospedagem (fornecendo destino (nome da cidade ou do hotel), data da entrada e data da saída, número de quartos, número e idade das pessoas e dados do cartão juntamente com a opção de parcelamento (se for efetuar a compra)).

Observações:

- Desenvolva uma interface gráfica com recursos de interação apropriados;
- Gerar a documentação completa de todas as classes e métodos de sua aplicação;
- Equipe: dois programadores ou individual.

2 - Desenvolvimento do projeto

Para a elaboração do código foi utilizada a linguagem de programação Javascript para a implementação do servidor. e Php, html e css para implementação do cliente da aplicação rest. O banco de dados usado foi o postgres carregado e configurado com a ajuda do docker-compose.

2.1 - Cliente

Na figura 01 temos os arquivos php que são usados para gerar a interface gráfica web escrita em php, html e css. o sistema do cliente no geral tem 4 telas principais:

1. comprar_hospedagem_page.php (compra e visualização de hospedagens compradas)
2. comprar_passagem_page.php (compra e visualização de passagens compradas)
3. consulta_hospedagem_disponivel.php (consultar hospedagens disponíveis)
4. consulta_passagens_disponivel.php (consultar passagens disponíveis)

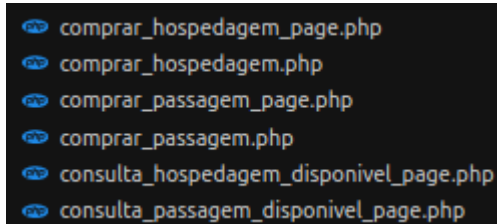
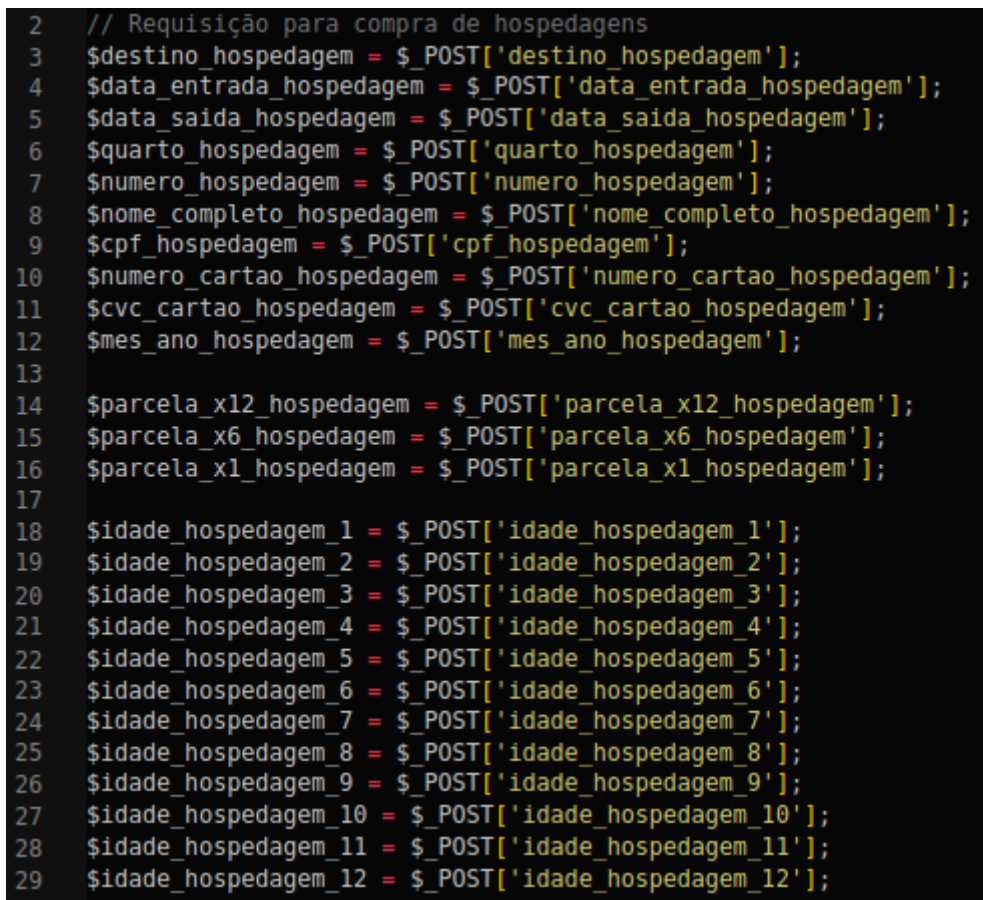


Figura 01: Arquivos do cliente.

Da linha 2 até a linha 29 temos os recebimentos dos dados enviados do html para o php.



```
2 // Requisição para compra de hospedagens
3 $destino_hospedagem = $_POST['destino_hospedagem'];
4 $data_entrada_hospedagem = $_POST['data_entrada_hospedagem'];
5 $data_saida_hospedagem = $_POST['data_saida_hospedagem'];
6 $quarto_hospedagem = $_POST['quarto_hospedagem'];
7 $numero_hospedagem = $_POST['numero_hospedagem'];
8 $nome_completo_hospedagem = $_POST['nome_completo_hospedagem'];
9 $cpf_hospedagem = $_POST['cpf_hospedagem'];
10 $numero_cartao_hospedagem = $_POST['numero_cartao_hospedagem'];
11 $cvc_cartao_hospedagem = $_POST['cvc_cartao_hospedagem'];
12 $mes_ano_hospedagem = $_POST['mes_ano_hospedagem'];
13
14 $parcela_x12_hospedagem = $_POST['parcela_x12_hospedagem'];
15 $parcela_x6_hospedagem = $_POST['parcela_x6_hospedagem'];
16 $parcela_x1_hospedagem = $_POST['parcela_x1_hospedagem'];
17
18 $idade_hospedagem_1 = $_POST['idade_hospedagem_1'];
19 $idade_hospedagem_2 = $_POST['idade_hospedagem_2'];
20 $idade_hospedagem_3 = $_POST['idade_hospedagem_3'];
21 $idade_hospedagem_4 = $_POST['idade_hospedagem_4'];
22 $idade_hospedagem_5 = $_POST['idade_hospedagem_5'];
23 $idade_hospedagem_6 = $_POST['idade_hospedagem_6'];
24 $idade_hospedagem_7 = $_POST['idade_hospedagem_7'];
25 $idade_hospedagem_8 = $_POST['idade_hospedagem_8'];
26 $idade_hospedagem_9 = $_POST['idade_hospedagem_9'];
27 $idade_hospedagem_10 = $_POST['idade_hospedagem_10'];
28 $idade_hospedagem_11 = $_POST['idade_hospedagem_11'];
29 $idade_hospedagem_12 = $_POST['idade_hospedagem_12'];
```

Figura 02: Recebendo dados do forms em php para compra de hospedagem.

Na figura 03 temos:

- Da linha 62 até a linha 76 temos a conversão dos dados para array e posteriormente para json.
- Na linha 78 é configurado a url para requisição POST do servidor.
- Da linha 80 até a linha 86 temos a configuração de uma requisição POST com body do tipo json, isso é feito com a extensão curl do php.
- Na linha 89 é feito o envio da requisição e capturado o seu retorno.

```
61
62 $array = array(
63     "destino_hospedagem"=> $destino_hospedagem,
64     "data_entrada_hospedagem"=> $data_entrada_hospedagem,
65     "data_saida_hospedagem"=> $data_saida_hospedagem,
66     "quarto_hospedagem"=> $quarto_hospedagem,
67     "numero_hospedagem"=> $numero_hospedagem,
68     "idade_hospedagem"=> $idades_hospedagem,
69     "nome_completo_hospedagem"=> $nome_completo_hospedagem,
70     "cpf_hospedagem"=> $cpf_hospedagem,
71     "numero_cartao_hospedagem"=> $numero_cartao_hospedagem,
72     "cvc_cartao_hospedagem"=> $cvc_cartao_hospedagem,
73     "mes_ano_hospedagem"=> $mes_ano_hospedagem,
74     "parcela_hospedagem"=> (int)$parcela_hospedagem
75 );
76 $json = json_encode($array);
77
78 $ch = curl_init('http://5b59b82e2666.ngrok.io/compra_hospedagem');
79
80 curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
81 curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
82 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
83 curl_setopt($ch, CURLOPT_HTTPHEADER, array(
84     'Content-Type: application/json',
85     'Content-Length: ' . strlen($json))
86 );
87
88 // Envio de requisição para compra de hospedagens
89 $jsonRet = json_decode(curl_exec($ch));
90
91 header('Location: comprar_hospedagem_page.'.php);
```

Figura 03: Configurando e enviando requisição para compra de hospedagem.

Na figura 04, da linha 2 até a linha 29 temos os recebimentos dos dados enviados do html para o php.

```
1  <?php
2  // Requisição para compra de passagens
3  $ida_passagem = $_POST['ida_passagem'];
4  $idaevolta_passagem = $_POST['idaevolta_passagem'];
5  $origem_passagem = $_POST['origem_passagem'];
6  $destino_passagem = $_POST['destino_passagem'];
7  $data_ida_passagem = $_POST['data_ida_passagem'];
8  $data_volta_passagem = $_POST['data_volta_passagem'];
9  $numero_passagem = $_POST['numero_passagem'];
10 $nome_completo_passagem = $_POST['nome_completo_passagem'];
11 $cpf_passagem = $_POST['cpf_passagem'];
12 $numero_cartao_passagem = $_POST['numero_cartao_passagem'];
13 $cvv_cartao_passagem = $_POST['cvv_cartao_passagem'];
14 $mes_ano_passagem = $_POST['mes_ano_passagem'];
15
16 $parcela_x12_passagem = $_POST['parcela_x12_passagem'];
17 $parcela_x6_passagem = $_POST['parcela_x6_passagem'];
18 $parcela_x1_passagem = $_POST['parcela_x1_passagem'];
19
20 $idade_passagem_1 = $_POST['idade_passagem_1'];
21 $idade_passagem_2 = $_POST['idade_passagem_2'];
22 $idade_passagem_3 = $_POST['idade_passagem_3'];
23 $idade_passagem_4 = $_POST['idade_passagem_4'];
24 $idade_passagem_5 = $_POST['idade_passagem_5'];
25 $idade_passagem_6 = $_POST['idade_passagem_6'];
26 $idade_passagem_7 = $_POST['idade_passagem_7'];
27 $idade_passagem_8 = $_POST['idade_passagem_8'];
28 $idade_passagem_9 = $_POST['idade_passagem_9'];
29 $idade_passagem_10 = $_POST['idade_passagem_10'];
30 $idade_passagem_11 = $_POST['idade_passagem_11'];
31 $idade_passagem_12 = $_POST['idade_passagem_12'];
```

Figura 04: Recebendo dados do forms em php para compra de passagem.

Na figura 05 temos:

- e. Da linha 77 até a linha 93 temos a conversão dos dados para array e posteriormente para json.
- f. Na linha 95 é configurado a url para requisição POST do servidor.
- g. Da linha 97 até a linha 103 temos a configuração de uma requisição POST com body do tipo json, isso é feito com a extensão curl do php.
- h. Na linha 106 é feito o envio da requisição e capturado o seu retorno.

```
77 $array = array(
78     "ida_passagem"=> $ida_passagem,
79     "idaevolta_passagem"=> $idaevolta_passagem,
80     "origem_passagem"=> $origem_passagem,
81     "destino_passagem"=> $destino_passagem,
82     "data_ida_passagem"=> $data_ida_passagem,
83     "data_volta_passagem"=> $data_volta_passagem,
84     "numero_passagem"=> $numero_passagem,
85     "nome_completo_passagem"=> $nome_completo_passagem,
86     "cpf_passagem"=> $cpf_passagem,
87     "numero_cartao_passagem"=> $numero_cartao_passagem,
88     "cvv_cartao_passagem"=> $cvv_cartao_passagem,
89     "mes_ano_passagem"=> $mes_ano_passagem,
90     "idade_passagem"=> $idades_passagem,
91     "parcela_passagem"=> (int)$parcela_passagem
92 );
93 $json = json_encode($array);
94
95 $ch = curl_init('http://5b59b82e2666.ngrok.io/compra_passagem/');
96
97 curl_setopt($ch, CURLOPT_CUSTOMREQUEST, "POST");
98 curl_setopt($ch, CURLOPT_POSTFIELDS, $json);
99 curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
100 curl_setopt($ch, CURLOPT_HTTPHEADER, array(
101     'Content-Type: application/json',
102     'Content-Length: ' . strlen($json))
103 );
104
105 // Requisição para compra de passagens
106 $jsonRet = json_decode(curl_exec($ch));
107
108 header('Location: comprar_passagem_page.'.php);
109
```

Figura 05: Configurando e enviando requisição para compra de passagem.

Na figura 06 da linha 257 à linha 264 temos a requisição GET em php para hospedagens compradas.

```
257 var data = JSON.parse('<?php
258     // echo ' Cidade pesquisada: ', $_POST['pesquisar_hospedagem'];
259     // echo '<br>';
260     // echo ' resultado: ';
261     // $pesquisar_hospedagem = $_POST['pesquisar_hospedagem'];
262     $data = file_get_contents('http://5b59b82e2666.ngrok.io/hospedagens_compradas/');
263     echo $data;
264 ?>')
265 console.log(data)
```

Figura 06: Fazendo requisição GET de hospedagens compradas.

Na figura 07 da linha 107 à linha 114 temos a requisição GET em php para passagens compradas.



```
107     var data = JSON.parse('<?php
108         // echo ' Cidade pesquisada: ';
109         // echo '<br>';
110         // echo 'resultado: ', $_POST['pesquisar_passagem'];
111         // $pesquisar_passagem = $_POST['pesquisar_passagem'];
112         $data = file_get_contents('http://5b59b82e2666.ngrok.io/todas_passagens_disponiveis');
113         echo $data;
114     ?>')
```

Figura 07: Fazendo requisição GET de passagens compradas.

Na figura 08 temos a tela de consulta das passagens disponíveis.

Destino	Quantidade
ceara	5
natal	8
rio de janeiro	45
sao paulo	30

Figura 08: Tela de consulta passagens disponíveis.

Na figura 09 temos a tela de compra e consulta de passagens compradas

Destino	Origem	Ida e volta	Data ida	Data volta	Número de pessoas	Idade das pessoas	Nome completo comprador	Cpf	Número do cartão	CVV cartão	Mes/Ano	Parcelas
Arabia	Marabá	false	2021-07-20	2021-07-30	2	1;2;	Cicrano	0000000	0000	0000	12/04	1
fortaleza	marabá	true	09/09/2021	[null]			Jaqueline Lima	000.000.000-00	54864323486	000	09/2022	12
istambul	marabá	false	2021-07-24	2021-08-07	4	27,33,37,34	Cicrano 3	00000000	2344		09/12	12
Lugar nenhum	Lugar algum	false	2021-08-06	2021-08-07	5	1;2;3;4;5;	Meu nome	123456789	111	1111	11/03	12
Moscow	Marabá	false	2021-07-30	2021-08-07	1	23	Iago Costa	0000000	999		06/12	6
Palmas	Cuiabá	false	2021-07-30	2021-08-06	4		Germanio	999999999	22222	22222	2222	12
Palmas	Cuiabá	false	2021-07-30	2021-08-06	1		Germanio	999999999	22222	22222	2222	6
Parauapebas	Itupiranga	false	2021-07-30	2021-08-06	1		Cicrano	999999999	22222	22222	2222	1
Rio de Janeiro	Marabá	false	2021-07-24	2021-08-04	1	21	Cicrano	0000000	189	4567	09/24	12

Figura 09: Tela de compra de passagem e consulta de passagens compradas.



Na figura 10 temos a tela de consulta de hospedagem disponível.

localhost/front-end/consulta_hospedagem_disponivel_page.php 80% Search

Comprar hospedagens Comprar passagens

Consulta de hospedagem disponível

Pesquisar

Column visibility Copy CSV Print Search:

Destino	Quantidade
ceara	8
natal	15

Showing 1 to 2 of 2 entries Previous 1 Next

Figura 10: Tela de consulta hospedagens disponíveis.

Na figura 11 temos a tela de compra e consulta de hospedagens compradas.

localhost/front-end/comprar_hospedagem_page.php 70% Search

Compra de hospedagem

Número de pessoas ok

Lugar de destino

Data de início da hospedagem mm/dd/yyyy Data de saída da hospedagem mm/dd/yyyy

Número de quartos

Nome Completo comprador CPF comprador

Número do cartão Código CVV do cartão Mes e Ano de Vencimento

Parcelamento ☐ x12 ☐ x1 ☐ x6

Comprar Hospedagem

Ver passagens disponíveis Ver hospedagens disponíveis

Consulta de hospedagens compradas

Ver hospedagens

Column visibility Copy CSV Print Search:

Destino	Data entrada	Data saída	Número de quartos	Número de pessoas	Idade das pessoas	Nome completo comprador	Cpf	Número do cartão	CVV cartão	Mes/Ano	Parcelas
Araguaina	2021-07-21	2021-08-06	3	3	45,23,17	Fulano de tal	0000000	0000	00	09/21	12
Araguaina	2021-07-22	2021-08-20	3	3	23,56,22	Fulano de tal	0000000	0000	00	09/21	6
Araguaina	2021-07-22	2021-08-20	3	2	45,23	Fulano de tal	0000000	0000	00	09/21	6
ceara	08/06/2022	12/06/2022	1	0002	25	Julia Gomes Fernandes	000.000.000-00	44595525611355	000	06/2022	6
Marabá	2021-08-05	2021-08-07	3	3	45,67,90	Cicrano de tal	0000000	0000	000	000000	1
Natal	05/06/2022	9/06/2022	1	0002	25	Julia Gomes Fernandes	000.000.000-00	44595525611355	000	06/2022	6
Natal	2021-07-22	2021-07-31	4	4	46,45,21,20	Fulano	000000	345	147	09/28	6
nova york	2021-07-04	2021-07-08	3	6	1;2;4;5;6	Cicrano	22222222	12222		02/14	17
rio de janeiro	2021-07-01	2021-07-09	2	2	1;2;	Fulano	0234034034	1232		02/14	12
sao paulo	2021-07-01	2021-07-09	2	2	1;2;	Fulano	0234034034	1232		02/14	12

Showing 1 to 10 of 10 entries Previous 1 Next

Figura 11: Tela de compra de hospedagem e consulta de hospedagens compradas.

2.2 – Servidor

Para a parte do servidor/back do projeto, foram utilizadas as seguintes tecnologias: Node.js, Express.js, TypeORM e Docker. Uma api feita em Node.js e utilizando o Express.js como middleware, foi desenvolvida para realizar a comunicação com o front.

Dentro da pasta src, que faz parte da estrutura do projeto, foi criada uma pasta chamada de entity, onde as tabelas do banco de dados foram especificadas utilizando a sintaxe do TypeORM, ao todo foram criadas 4 tabelas para o projeto, sendo elas: compra_passagem, compra_hospedagem, hospedagens_disponiveis, e passagens_disponiveis. Nas imagens abaixo, é possível observar a estrutura da tabela compra_passagem, que segue no mesmo padrão para as demais tabelas.

```
1 import {Entity, PrimaryGeneratedColumn, Column} from "typeorm";
2
3 @Entity()
4 export class Compra_passagem {
5     constructor(
6         ida_passagem: boolean, idaevolta_passagem: boolean, origem_passagem: string,
7         destino_passagem: string, data_ida_passagem: string, data_volta_passagem: string,
8         nome_completo_passagem: string, cpf_passagem: string, numero_cartao_passagem: string,
9         cvv_cartao_passagem: string, mes_ano_passagem: string, parcela_passagem: number,
10        numero_passagem: string, idade_passagem: string){
11
12        this.ida_passagem = ida_passagem;
13        this.idaevolta_passagem = idaevolta_passagem;
14        this.origem_passagem = origem_passagem;
15        this.destino_passagem = destino_passagem;
16        this.data_ida_passagem = data_ida_passagem;
17        this.data_volta_passagem = data_volta_passagem;
18        this.nome_completo_passagem = nome_completo_passagem;
19        this.cpf_passagem = cpf_passagem;
20        this.numero_cartao_passagem = numero_cartao_passagem;
21        this.cvv_cartao_passagem = cvv_cartao_passagem;
22        this.mes_ano_passagem = mes_ano_passagem;
23        this.parcela_passagem = parcela_passagem;
24        this.numero_passagem = numero_passagem;
25        this.idade_passagem = idade_passagem;
26    }
```

Figura 12: Estrutura da tabela compra_passagem parte-1

```
28     @PrimaryGeneratedColumn()
29     id: number;
30
31     @Column({nullable: true})
32     ida_passagem: boolean; @Column({nullable: true})
33     idaevolta_passagem: boolean;
34
35     @Column({nullable: true})
36     origem_passagem: string;
37
38     @Column()
39     destino_passagem: string;
40
41     @Column({nullable: true})
42     data_ida_passagem: string;
43
44     @Column({nullable: true})
45     data_volta_passagem: string;
46
47     @Column({nullable: true})
48     nome_completo_passagem: string;
49
50     @Column({nullable: true})
51     cpf_passagem: string;
52
53     @Column({nullable: true})
54     numero_cartao_passagem: string;
```

Figura 13: Estrutura da tabela compra_passagem parte-2

```
56     @Column({nullable: true})
57     cvv_cartao_passagem: string;
58
59     @Column({nullable: true})
60     mes_ano_passagem: string;
61
62     @Column({nullable: true})
63     parcela_passagem: number;
64
65     @Column({nullable: true})
66     numero_passagem: string
67
68     @Column({nullable: true})
69     idade_passagem: string
70
71 }
72
```

Figura 14 : Estrutura da tabela compra_passagem parte-3

O docker foi utilizado para fazer a containerização do banco de dados, assim foi criado um container onde todas as configurações para o funcionamento do banco de dados, que no caso do projeto foi o Postgres, foram descritas. Essas configurações são descritas em um arquivo chamado de docker-compose. A estrutura do arquivo pode ser vista na imagem abaixo:

```
1  version: '3'
2  services:
3    db:
4      image: 'postgres:latest'
5      environment:
6        POSTGRES_PASSWORD: postgres
7        POSTGRES_USER: postgres
8        POSTGRES_DB: appProjetoSdAPI
9      volumes:
10       - ./pgdata:/varlib/postgresql/data
11      networks:
12       - postgres-compose-network
13      ports:
14       - "5432:5432"
15
16     teste-pgadmin-compose:
17       image: dpage/pgadmin4
18       environment:
19         PGADMIN_DEFAULT_EMAIL: "juliana.batista@unifesspa.edu.br"
20         PGADMIN_DEFAULT_PASSWORD: "admin"
21       ports:
22       - "16543:80"
23       depends_on:
24       - db
25       networks:
26       - postgres-compose-network
27
28   networks:
29     postgres-compose-network:
30       driver: bridge
```

Figura 15: Estrutura do arquivo docker-compose

Os métodos para manipular os dados de cada tabela, são definidos em um arquivo denominado de Controller para cada uma das tabelas existentes. Abaixo se encontra a estrutura do arquivo `Compra_passagemController`, referente a tabela `compra_passagem`.

```
1  import { getManager } from "typeorm";
2  import { runInThisContext } from "vm";
3  import { Compra_passagem } from "../entity/Compra_passagem";
4
5  export class Compra_passagemController{
6      async salvar(passagem_comprada: Compra_passagem){
7          const passagem_comprada_salva = await getManager().save(passagem_comprada);
8          return passagem_comprada_salva;
9      }
10
11     async recuperaPorId(id: number){
12         const passagem_comprada = await getManager().findOne(Compra_passagem, id);
13         return passagem_comprada;
14     }
15
16     async recuperaTodas(){
17         const passagens_compradas = await getManager().find(Compra_passagem);
18         return passagens_compradas;
19     }
20 }
```

Figura 16: Estrutura do arquivo `Compra_passagemController`

Os métodos definidos no Controller são então utilizados no arquivo de rota correspondente a cada tabela, esse arquivo possui o mesmo nome da tabela em questão, onde são definidas as rotas que serão utilizadas para fazer as requisições para a api. Abaixo, é possível observar a estrutura do arquivo de rota da tabela compra_passagem.

```
1 import { Router } from "express";
2 import { Compra_passagemController } from "../controller/Compra_passagemController";
3 import { Compra_passagem } from "../entity/Compra_passagem";
4
5 export const routerCompra_passagem = Router();
6 export const routerPassagens_compradas = Router();
7 export const routerPassagem_comprada_numero = Router();
8 const compra_passagemCtrl = new Compra_passagemController();
9
10 routerCompra_passagem.post('/', async (req, res) => {
11   const { ida_passagem, idaevolta_passagem, origem_passagem, destino_passagem, data_ida_passagem, data_volta_passagem,
12     nome_completo_passagem, cpf_passagem, numero_cartao_passagem, cvv_cartao_passagem, mes_ano_passagem, parcela_passagem,
13     numero_passagem, idade_passagem } = req.body;
14   const compra_passagem = new Compra_passagem([ida_passagem, idaevolta_passagem, origem_passagem, destino_passagem,
15     data_ida_passagem, data_volta_passagem,
16     nome_completo_passagem, cpf_passagem, numero_cartao_passagem, cvv_cartao_passagem, mes_ano_passagem,
17     parcela_passagem, numero_passagem, idade_passagem]);
18   const compra_passagem_salvo = await compra_passagemCtrl.salvar(compra_passagem);
19   res.json(compra_passagem_salvo);
20 });
21
22 routerCompra_passagem.get('/:idCompra_passagem', async (req, res) => {
23   const idCompra_passagem = parseInt (req.params.idCompra_passagem);
24   const compra_passagem_por_id = await compra_passagemCtrl.recuperaPorId(idCompra_passagem);
25   res.json(compra_passagem_por_id);
26 });
27
28
29 routerPassagens_compradas.get('/', async (req, res) => {
30   const todas_passagens_compradas = await compra_passagemCtrl.recuperaTodas();
31   res.json(todas_passagens_compradas);
```

Figura 17: Estrutura do arquivo de definição de rota da tabela compra_passagem

Por fim, no arquivo onde é criada a aplicação Express, as rotas para acesso da api são passadas junto a uma identificação que faz referência a respectiva rota. Assim, para que o front possa se comunicar com a api do projeto, é necessário utilizar as identificações que foram passadas no final da url base do projeto. Abaixo se encontra a estrutura do arquivo app.

```
1 import * as express from 'express';
2 import * as bodyParser from 'body-parser';
3 import * as cors from 'cors';
4 import * as logger from 'morgan';
5 import { conectarServidorNoBD } from './config/db';
6 import { routerCompra_hospedagem } from './routes/compra_hospedagem';
7 import { routerHospedagens_compradas } from './routes/compra_hospedagem';
8 import { routerHospedagem_comprada_numero } from './routes/compra_hospedagem';
9 import { routerPassagens_compradas } from './routes/compra_passagem';
10 import { routerCompra_passagem } from './routes/compra_passagem';
11 import { routerHospedagens_disponiveis } from './routes/hospedagens_disponiveis';
12 import { routerTodas_hospedagens_disponiveis } from './routes/hospedagens_disponiveis';
13 import { routerPassagens_disponiveis } from './routes/passagens_disponiveis';
14 import { routerTodas_passagens_disponiveis } from './routes/passagens_disponiveis';
15
16 export const app = express();
17
18 app.use(cors());
19
20 app.use(bodyParser.json());
21
22 app.use(logger('dev'));
23
24 conectarServidorNoBD();
25
26 app.use('/compra_hospedagem', routerCompra_hospedagem);
27 app.use('/hospedagens_compradas', routerHospedagens_compradas);
28 app.use('/hospedagem_comprada_numero', routerHospedagem_comprada_numero);
29 app.use('/passagens_compradas', routerPassagens_compradas);
30 app.use('/compra_passagem', routerCompra_passagem);
31 app.use('/hospedagens_disponiveis', routerHospedagens_disponiveis);
32 app.use('/todas_hospedagens_disponiveis', routerTodas_hospedagens_disponiveis);
33 app.use('/passagens_disponiveis', routerPassagens_disponiveis);
34 app.use('/todas_passagens_disponiveis', routerTodas_passagens_disponiveis);
35 app.use('/', (req, res) => res.send(
36   'Api do app para o projeto de Sistemas Distribuidos'
37 ));
```

Figura 18: Estrutura do arquivo app

3 - Conclusão

Ao longo do desenvolvimento do código, algumas dificuldades foram encontradas, dentre elas podemos destacar a falta de experiência do desenvolvedor com tratamento de dados em php e utilização da extensão cURL, com isso foi necessário visitar documentações e fóruns para conseguir concluir os scripts de requisição post e get na parte do cliente.

Mesmo com alguns erros e dificuldades na implementação do trabalho, os pontos presentes no roteiro foram atendidos, sendo o código capaz de executar o que foi inicialmente proposto no documento.

4 - Referências

PHP: cURL. **Php:cURL - Manual.** Disponível em:
https://www.php.net/manual/pt_BR/book.curl.php. Acesso em: 15 jul. 2021.