Linha 15 classe FuncionarioService:

```java
import com.senai.jonatas.funcionarios.dto.FuncionarioRequest;
import com.senai.jonatas.funcionarios.dto.FuncionarioResponse;
import com.senai.jonatas.funcionarios.exceptions.*;
import com.senai.jonatas.funcionarios.mapper.FuncionarioMapper;
import com.senai.jonatas.funcionarios.entity.Funcionario;
import com.senai.jonatas.funcionarios.repository.FuncionarioRepository;
import jakarta.transaction.Transactional;
import org.springframework.stereotype.Service;

import java.time.LocalDate;
import java.util.List;

@Service
public class FuncionarioService {
```

```java
    @Transactional
    public Result<FuncionarioResponse> cadastrar(FuncionarioRequest req) {
        validarRegrasComuns(req);
        var existenteOpt = repository.findByEmailIgnoreCase(req.email());

        if (existenteOpt.isPresent()) {
            var existente = existenteOpt.get();
            if (Boolean.TRUE.equals(existente.getAtivo())) {
                throw new EmailConflictException("E-mail já cadastrado e ativo");
            }
            FuncionarioMapper.copyToEntity(req, existente);
            existente.setAtivo(true);
            var salvo = repository.save(existente);
            return Result.reactivated(FuncionarioMapper.toResponse(salvo));
        }
        var novo = FuncionarioMapper.toEntity(req);
        var salvo = repository.save(novo);
        return Result.created(FuncionarioMapper.toResponse(salvo));
    }

    @Transactional
    public FuncionarioResponse atualizar(Long id, FuncionarioRequest req) {
        validarRegrasComuns(req);

        var existente = repository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Funcionário não encontrado: " + id));
        if (!req.email().equalsIgnoreCase(existente.getEmail())) {
            if (repository.existsByEmailIgnoreCase(req.email())) {
                throw new EmailConflictException("E-mail já está sendo usado por outro funcionário");
            }
        }
        FuncionarioMapper.copyToEntity(req, existente);
        var salvo = repository.save(existente);
        return FuncionarioMapper.toResponse(salvo);
    }
```

```java
    @Transactional
    public FuncionarioResponse inativar(Long id) {
        var existente = repository.findById(id)
                .orElseThrow(() -> new ResourceNotFoundException("Funcionário não encontrado"));
        if (Boolean.FALSE.equals(existente.getAtivo())) {
            throw new BusinessException("Funcionário já está inativo");
        }
        existente.setAtivo(false);
        var salvo = repository.save(existente);
        return FuncionarioMapper.toResponse(salvo);
    }
```

Linha 9 classe FuncionarioRepository:

```java
public interface FuncionarioRepository extends JpaRepository<Funcionario, Long> {
```

Classe da entidade funcionário:

```java
@Entity
@Table(name = "funcionarios", uniqueConstraints = {
        @UniqueConstraint(name = "uk_funcionario_email", columnNames = "email")
})
public class Funcionario {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "nome", nullable = false)
    private String nome;

    @Column(name = "email", nullable = false)
    private String email;

    @Column(name = "cargo", nullable = false)
    private String cargo;

    @Column(name = "salario", nullable = false)
    private BigDecimal salario;

    @Column(name = "data_admissao", nullable = false)
    private LocalDate dataAdmissao;

    @Column(name = "ativo", nullable = false)
    private Boolean ativo;
```

Classe FuncionarioController:

```java
package com.senai.jonatas.funcionarios.controller;

import com.senai.jonatas.funcionarios.dto.FuncionarioRequest;
import com.senai.jonatas.funcionarios.dto.FuncionarioResponse;
import com.senai.jonatas.funcionarios.service.FuncionarioService;
import jakarta.validation.Valid;
import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.util.UriComponentsBuilder;

import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;

@RestController
@RequestMapping("/api/funcionarios")
@CrossOrigin(origins = "http://localhost:4200")
public class FuncionarioController {

    @Autowired
    private FuncionarioService service;
```

Mapper:

```java
public static Funcionario toEntity(FuncionarioRequest req) {
    return Funcionario.builder()
            .nome(req.nome())
            .email(req.email())
            .cargo(req.cargo())
            .salario(req.salario())
            .dataAdmissao(req.dataAdmissao())
            .ativo(true)
            .build();
}
```

Application.propriets:

```
spring.application.name=funcionarios

info.app.description=@project.description@
info.app.encoding=@project.build.sourceEncoding@
info.build.artifact=@project.artifactId@
info.build.group=@project.groupId@
info.build.name=@project.name@
info.build.version=@project.version@

spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
spring.datasource.url=jdbc:h2:file:./data/db-api;DB_CLOSE_ON_EXIT=FALSE
spring.h2.console.enabled=true
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

Pom.xml:

```
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
</dependency>
```

Frontend, isso acontece também nas outras interfaces, que seria o nome esta number:

```
export interface FuncionarioRequest {
  nome: string;
  email: string;
  cargo: string;
  salario: number;
  dataAdmissao: string; // ISO
}
```

Funcionario-forms:

```
// Componentes PrimeNG
import { ToastModule } from 'primeng/toast';
import { ButtonModule } from 'primeng/button';
import { InputTextModule } from 'primeng/inputtext';
import { InputNumberModule } from 'primeng/inputnumber';
import { CalendarModule } from 'primeng/calendar';

import { FuncionarioRequest } from "../../models/funcionarioRequest";

@Component({
  selector: 'app-funcionario-form',
  standalone: true,
  imports: [
    FormsModule,
    RouterLink,
    ToastModule,
    ButtonModule,
    InputTextModule,
    InputNumberModule,
    CalendarModule,
  ],
  templateUrl: './funcionario-form.component.html',
  styleUrls: ['./funcionario-form.component.css']
})
export class FuncionarioFormComponent implements OnInit {
  id: number | null = null;
```

```
  dataAdmissaoModel: Date | string = new Date();
  toDate: Date = new Date();

  constructor(
    private service: FuncionarioService,
    private route: ActivatedRoute,
    private router: Router,
    private msg: MessageService
  ) { }
  ngOnInit(): void {
    const paramId = this.route.snapshot.paramMap.get('id');
    if (paramId) {
      this.isEdicao = true;
      this.id = Number(paramId);
      this.carregarFuncionario(this.id);
    } else {
      this.dataAdmissaoModel = new Date();
    }
  }
```

```
        };
        this.dataAdmissaoModel = new Date(f.dataAdmissao + 'T00:00:00');
        this.service.loading.set(false);
      },
      error: () => {
        this.msg.add({ severity: 'error', summary: 'Erro', detail: 'Funcionário não encontrado' });
        this.service.loading.set(false);
        this.router.navigate(['/funcionarios']);
      }
    });
  }
  prepararParaSalvar() {
    if (this.dataAdmissaoModel instanceof Date) {
      const date = this.dataAdmissaoModel;
      const year = date.getFullYear();
      const month = String(date.getMonth() + 1).padStart(2, '0');
      const day = String(date.getDate()).padStart(2, '0');
      this.funcionario.dataAdmissao = `${year}-${month}-${day}`;
    }
  }

  salvar() {
    this.prepararParaSalvar();
    if (!this.validarCampos()) return;
```

```
    this.dataAdmissaoModel = new Date();
    this.msg.add({ severity: 'info', summary: 'Info', detail: 'Campos limpos' });
    }
```

```
    if (f.salario <= 0) {
      this.msg.add({ severity: 'warn', summary: 'Validação', detail: 'Salário deve ser maior que zero' });
      return false;
    }

    if (!f.dataAdmissao || f.dataAdmissao > this.hojeISO()) {
      this.msg.add({ severity: 'warn', summary: 'Validação', detail: 'Data de admissão inválida' });
      return false;
    }
```

App.routes.ts:

src > app > TS app.routes.ts > ...

```
import { Routes } from '@angular/router';
import { FuncionarioListComponent } from "./components/funcionario-list/funcionario-list.component";
import { FuncionarioFormComponent } from "./components/funcionario-form/funcionario-form.component";

export const routes: Routes = [
  { path: '', redirectTo: 'funcionarios', pathMatch: 'full' },
  { path: 'funcionarios', component: FuncionarioListComponent },
  { path: 'funcionarios/novo', component: FuncionarioFormComponent },
  { path: 'funcionarios/editar/:id', component: FuncionarioFormComponent },
  { path: '**', redirectTo: '/funcionarios' }
];
```

FuncionarioService:

```
  buscarPorId(id: number): Observable<FuncionarioResponse> {
    return this.http.get<FuncionarioResponse>(`${this.baseUrl}/${id}`);
  }

  criar(req: FuncionarioRequest): Observable<FuncionarioResponse> {
    return this.http.post<FuncionarioResponse>(this.baseUrl, req);
  }

  atualizar(id: number, req: FuncionarioRequest): Observable<FuncionarioResponse> {
    return this.http.put<FuncionarioResponse>(`${this.baseUrl}/${id}`, req);
  }

  inativar(id: number): Observable<FuncionarioResponse> {
    return this.http.put<FuncionarioResponse>(`${this.baseUrl}/${id}/inativar`, {});
  }

  // helpers
  setCache(list: FuncionarioResponse[]) { this._cache$.next(list); }
  clearCache() { this._cache$.next(null); }
}

```

App.config:

```
import {ApplicationConfig, importProvidersFrom} from '@angular/core';
import {provideRouter} from '@angular/router';
import { provideHttpClient } from '@angular/common/http';
import {routes} from './app.routes';
import {BrowserAnimationsModule} from "@angular/platform-browser/animations";
import {ConfirmationService, MessageService} from "primeng/api";

export const appConfig: ApplicationConfig = {
  providers: [
    provideRouter(routes),
    importProvidersFrom(BrowserAnimationsModule),
    provideHttpClient(),
    MessageService,
    ConfirmationService
  ]
};
```

Environment:

```
src > environments > TS environment.ts > environment > apiUrl
  export const environment = {
    production: true,
    apiUrl: 'http:/localhost:8080/api'
  };

```

Html – forms:

```
    <div class="form-row">
      <label for="email">E-mail</label>
      <input pInputText id="email" [(ngModel)]="funcionario.email" placeholder="nome@dominio.com" />
    </div>

    <div class="form-row">
      <label for="cargo">Cargo</label>
      <input pInputText id="cargo" [(ngModel)]="funcionario.cargo" placeholder="Ex.: Analista" />
    </div>

    <div class="form-row">
      <label for="salario">Salário</label>
      <p-inputNumber id="salario" mode="currency" currency="BRL" locale="pt-BR"
        [(ngModel)]="funcionario.salario"></p-inputNumber>
    </div>

    <div class="form-row">
      <label for="dataAdmissao">Data de Admissão</label>
      <p-calendar id="dataAdmissao" dateFormat="yy-mm-dd" [maxDate]="toDate" [(ngModel)]="dataAdmissaoModel"></p-calendar>
    </div>

    <div class="actions">
      <button pButton label="Salvar" (click)="salvar()" [disabled]="carregando()"></button>
      <button pButton label="Limpar" class="p-button-secondary" (click)="limpar()" [disabled]="carregando()"></button>
      <a routerLink="/funcionarios">
        <button pButton label="Voltar" class="p-button-text"></button>
      </a>
    </div>
  </div>
```