



Inside Provider

Discentes: João Abreu, Lucas Spier, Willian Colombo

Introdução



Sistema de gerenciamento interno
para provedores de internet.

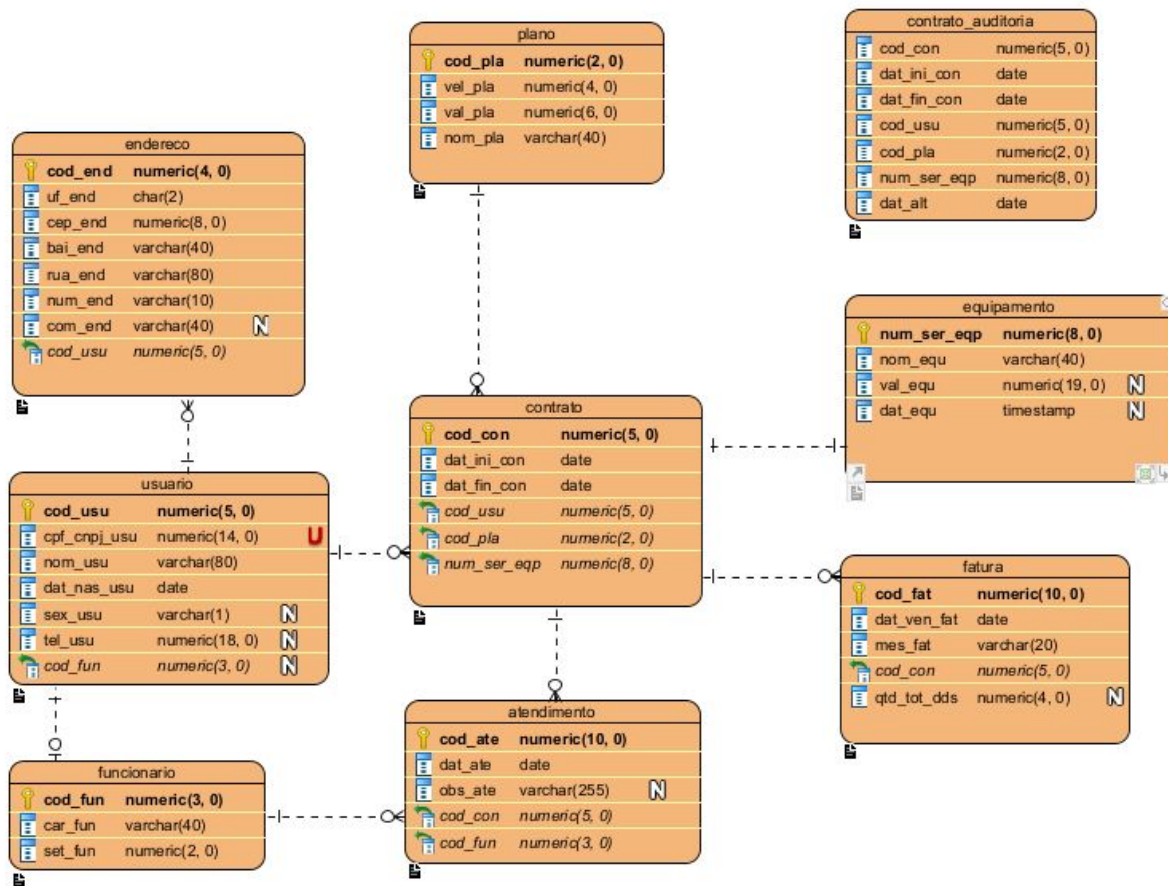


BANCO DE DADOS II

- Atualização do modelo relacional
- Atualização dos scripts de criação
- Criação dos índices
- Controle de acesso
- Criação das triggers
- Criação das funções
- Consultas
- Backup e restore

Modelo Relacional

- Exclusão da tabela “plano_equipamento”
- Adição da tabela “contrato_auditoria”
- Atualização dos *scripts*



Índices

- Pequena quantidade de dados
- Adicionados nas chaves-estrangeiras
- Adicionados nas colunas mais utilizadas

Usuário (CPF/CNPJ, telefone,)
Atendimento (Observações)
Equipamento (Nome)
Plano (Velociade)



Controle de acesso

ÁREAS DO NEGÓCIO					
Objetos de negocio	Comercial	Suporte	Financeiro	Administrador	
Endereço	C-A-V	A-V	V	T	
Usuario	C-A-V	A-V	V	T	Legenda
Funcionario	-	-	V	T	C - Cadastrar
Atendimento	V	V	V	T	E - Excluir
Contrato	C-A-V	V	V	T	A - Atualizar
Fatura	V	V	C-A-V	T	V - Visualizar
Plano	V	V	V	T	T - Todos

- Divisão do negócio em setores
- Atribuições de privilégios
- Criação de usuários com login

Triggers

```
CREATE FUNCTION verificaFatura() RETURNS trigger
AS $$
BEGIN

    IF NEW.cod_fat EXISTS THEN
        RAISE EXCEPTION 'Codigo Da Fatura Ja Existe';
    END IF;
    IF NEW.qtd_tot_dds IS NULL THEN
        RAISE EXCEPTION 'Quantidade nao pode ser nula';
    END IF;
    IF NEW.qtd_tot_dds < 1 THEN
        RAISE EXCEPTION 'Nao e possivel faturar clientes sem consumo de banda';
    END IF;

END;
$$
LANGUAGE plpgsql;

CREATE TRIGGER verificaFatura BEFORE INSERT OR UPDATE ON fatura
FOR EACH ROW EXECUTE FUNCTION verificaFatura();
```

Triggers

```
create table contrato_auditoria (  
    cod_con numeric(5) not null,  
    dat_ini_con date not null,  
    dat_fin_con date not null,  
    cod_usu numeric (5) not null,  
    cod_pla numeric (2) not null,  
    num_ser_eqp numeric (8) not null,  
    dat_alt Date not null  
);
```

```
create or replace trigger contrato_auditoria_trigger  
after insert or delete on contrato  
for each row  
execute procedure contrato_auditoria_function();  
  
create or replace function contrato_auditoria_function ()  
returns trigger as  
$$  
begin  
    insert into contrato_auditoria  
    (cod_con, dat_ini_con, dat_fin_con, cod_usu, cod_pla, num_ser_eqp, dat_alt)  
    values (new.cod_con, new.dat_ini_con, new.dat_fin_con, new.cod_usu, new.cod_pla, new.num_ser_eqp, current_date);  
    return new;  
end  
$$  
language plpgsql;
```


Funções

```
CREATE OR REPLACE FUNCTION getClientes() RETURNS SETOF usuario AS
$body$
DECLARE
    x usuario%rowtype;
BEGIN
    FOR x IN SELECT * FROM usuario

    LOOP
        IF (x.cod_fun is null) THEN
            RETURN NEXT x;
        END IF;
    END LOOP;

    RETURN;
END
$body$
LANGUAGE plpgsql;
```

Funções

```
CREATE OR REPLACE FUNCTION getUsuarioAtendimento(cpf numeric(14))
RETURNS table(cod_ate numeric, dat_ate date, obs_ate varchar) AS
$body$
BEGIN
    RETURN query select a.cod_ate, a.dat_ate, a.obs_ate from usuario u
        inner join contrato c on u.cod_usu = c.cod_usu
        inner join atendimento a on c.cod_con = a.cod_con
        where u.cpf_cnpj_usu = $1
        order by u.cod_usu;
END
$body$
LANGUAGE 'plpgsql';
```

Consultas



```
/* PRIMEIRO RELATÓRIO
Relacionar o código, nome e todos os clientes que são pessoa física.
Ordene o relatório de forma descendente pelo nome.
*/

create or replace view vw_relatorio1 as
select cod_usu "Código do usuário", nom_usu "Nome do usuário", cpf_cnpj_usu "CPF do usuário"
from usuario u
where length (cpf_cnpj_usu::varchar) < 13
order by nom_usu desc;
```

```
/* SEGUNDO RELATÓRIO
Relacionar o nome do cliente, nome do plano e a velocidade para todos os clientes.
Filtre somente clientes com planos com velocidades maiores que 100MB.
Ordene o relatório de forma descendente pelo nome do plano.
*/

create or replace view vw_relatorio2 as
select u.nom_usu "Nome do usuário", p.nom_pla "Nome do plano", p.vel_pla "Velocidade do plano"
from usuario u
inner join contrato c on u.cod_usu = c.cod_usu
inner join plano p on c.cod_pla = p.cod_pla
where p.vel_pla > 100
order by p.nom_pla desc;
```

Consultas



```
/* TERCEIRO RELATÓRIO
Relacionar o código do cliente, nome do cliente, quantidade total de
atendimentos nos meses pares de 2022. Ordene o relatório do cliente com mais
atendimentos(em termos de quantidade) para o cliente com menos atendimentos.
*/

create or replace view vw_relatorio3 as
select u.cod_usu "Código do usuário", u.nom_usu "Nome do usuário", count(a.cod_ate) "Quantidade de atendimentos"
  from usuario u
 inner join contrato c on u.cod_usu = c.cod_usu
 inner join atendimento a on c.cod_con = a.cod_con
 where mod(extract(month from a.dat_ate), 2) = 0
 group by u.cod_usu
 order by 3 desc;
```

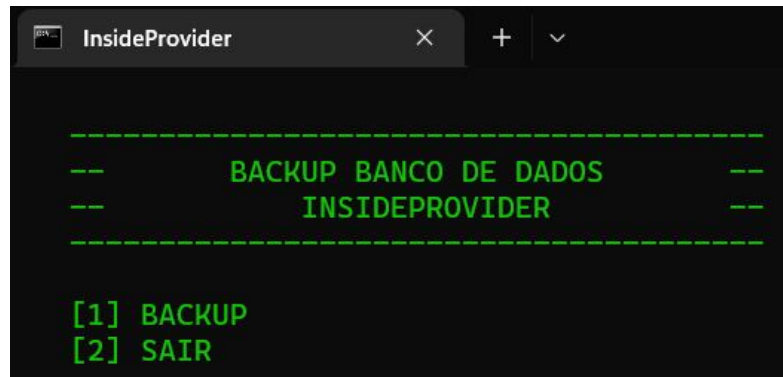
Consultas

```
/* QUARTO RELATÓRIO
Relacionar o cpf do cliente, nome do cliente e o valor total de uso da internet.
Filtrar somente clientes do sexo masculino, com idades pares e que realizaram compras em
meses entre 01 e 08 de 2022. Ordene o relatório do cliente com maior uso até o cliente com menor uso.
*/

create or replace view vw_relatorio4 as
select u.cpf_cnpj_usu "CPF do usuário", u.nom_usu "Nome do usuário", f.qtd_tot_dds "Quantidade de dados"
  from usuario u
 inner join contrato c on u.cod_usu = c.cod_usu
 inner join fatura f on c.cod_con = f.cod_con
 where u.sex_usu = 'M' and mod(extract(year from age(u.dat_nas_usu)), 2) = 0
 and extract(month from c.dat_ini_con) > 0 and extract(month from c.dat_ini_con) < 9
 and extract(year from c.dat_ini_con) = 2022
 order by f.qtd_tot_dds desc;
```

Backup e Restore

- *Script* semiautomático para backup e restore
- Executável .bat com menu



```
InsideProvider  X  +  v

-----
--          BACKUP BANCO DE DADOS          --
--          INSIDEPROVIDER                  --
-----

[1] BACKUP
[2] SAIR
```



```
InsideProvider  X  +  v

-----
--          RESTORE BANCO DE DADOS          --
--          INSIDEPROVIDER                  --
-----

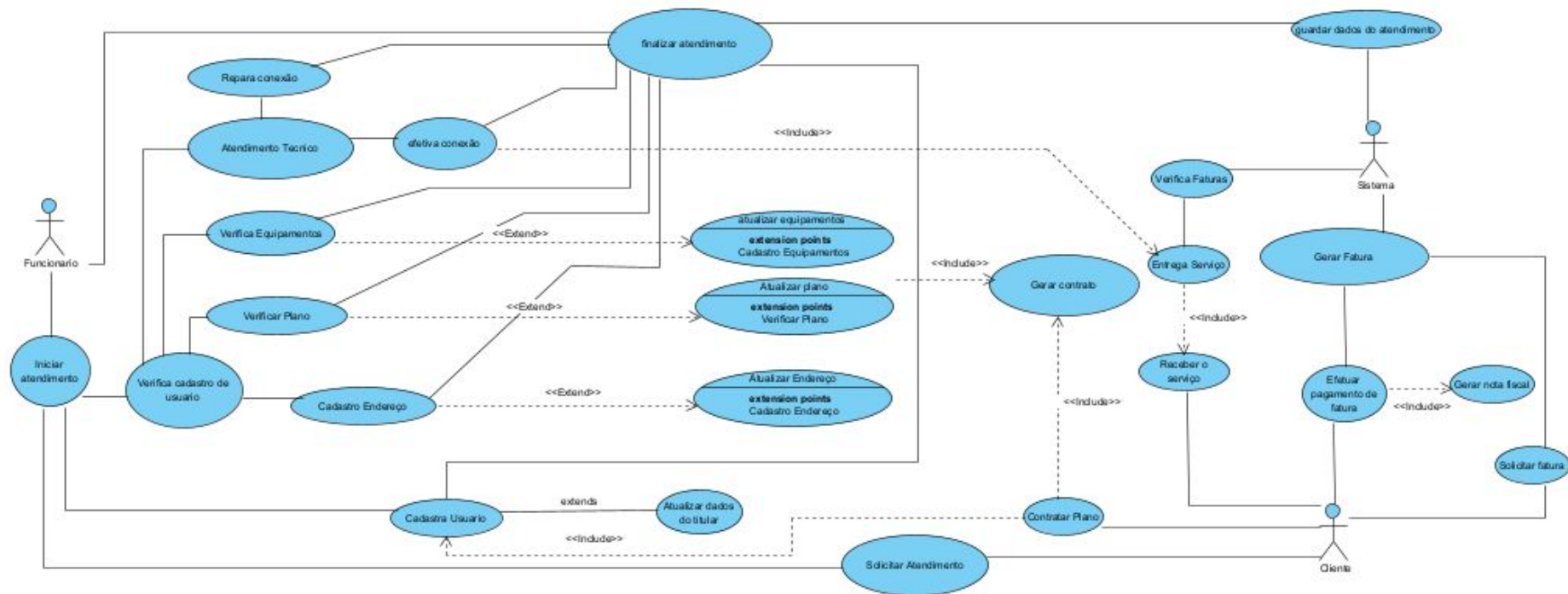
[1] RESTORE
[2] SAIR
```



ENGENHARIA DE SOFTWARE I

- Diagrama caso de uso
- Fluxo caso de uso
- Diagrama de atividades
- Diagrama de classes
- Diagrama de estados
- Diagrama de sequência

Diagrama de caso de uso



Fluxo de caso de uso



Fluxo de Caso de uso: atendimento

1. Cliente solicita atendimento
2. Funcionário inicia atendimento ao cliente
3. Funcionário verifica o cadastro do cliente
4. Cliente solicita algum serviço disponível, que pode ser: atendimento técnico, mudança de plano, equipamento ou de endereço
5. Funcionário abrirá o atendimento conforme o que foi solicitado pelo cliente (efetivar ou reparar conexão, atualização de plano, atualização de endereço ou atualização de equipamento)
6. Funcionário termina o atendimento conforme validado pelo cliente
7. Sistema Guarda os dados do atendimento.

Fluxo de Caso de uso: Faturamento

1. Cliente solicita fatura
2. Sistema gera fatura
3. Cliente recebe fatura
4. Cliente Paga a fatura
5. Sistema gera nota fiscal
6. Sistema verifica fatura e entrega serviço para o cliente correspondente ao mês pago

Diagrama de atividades

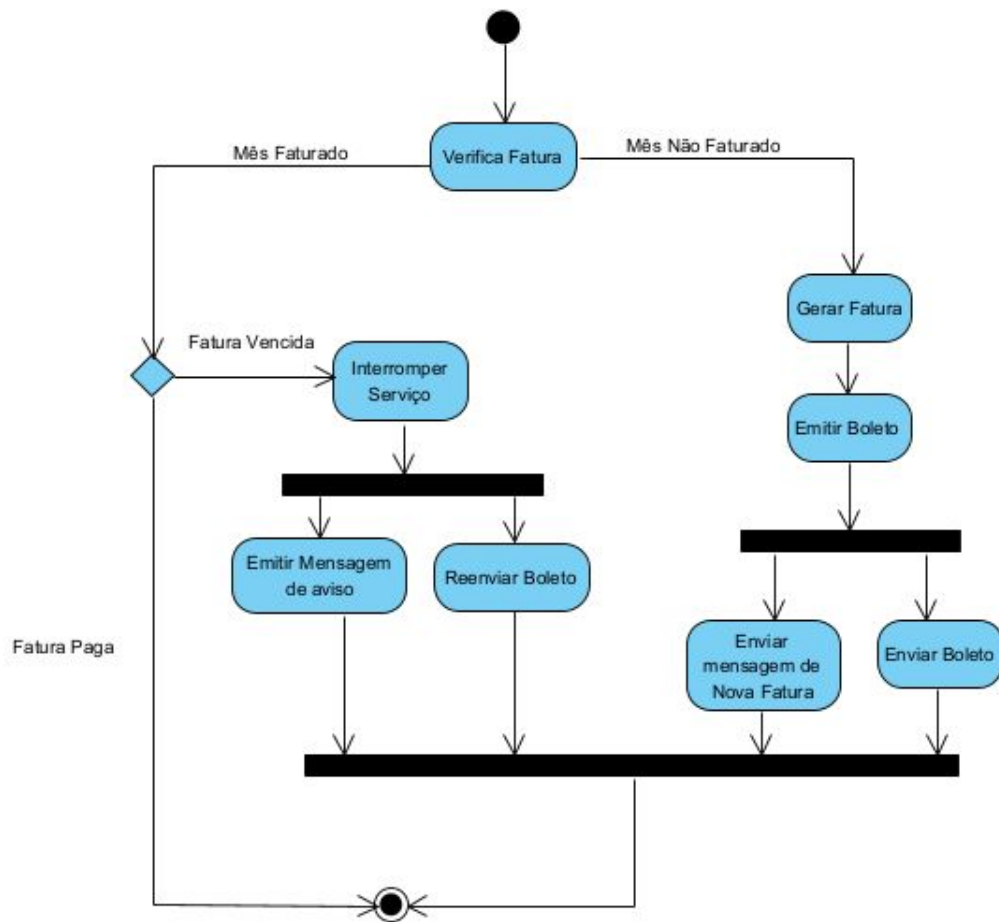


Diagrama de classes

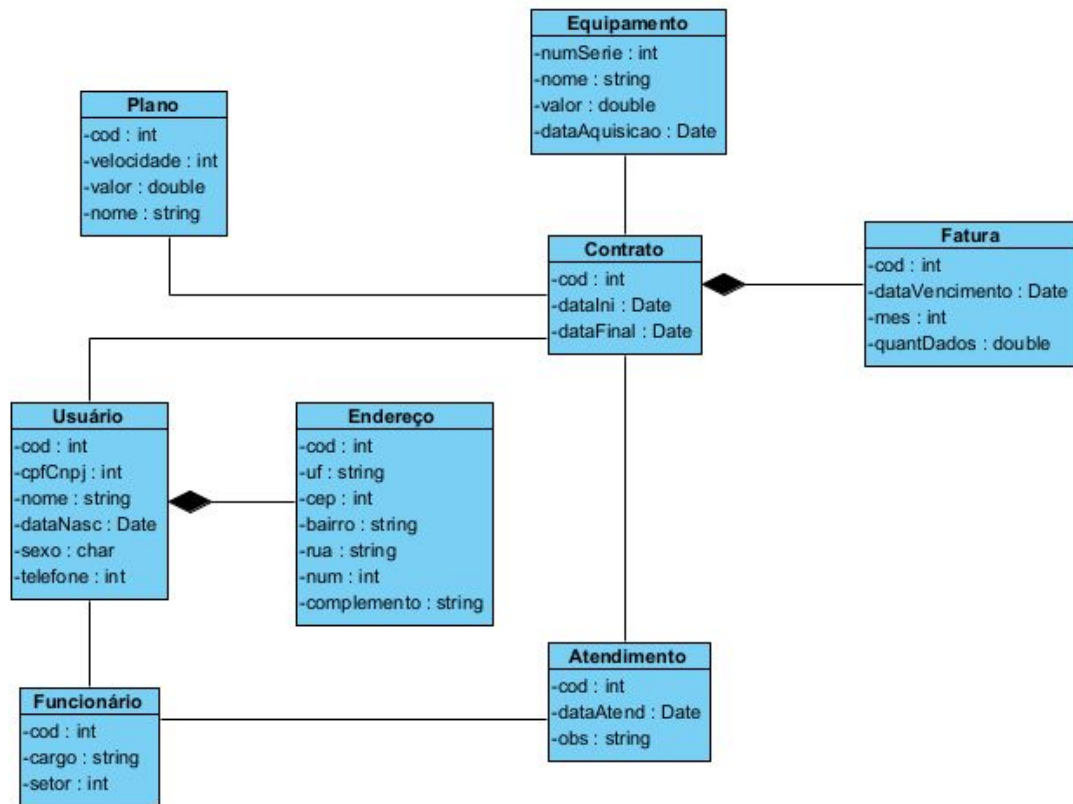


Diagrama de estados

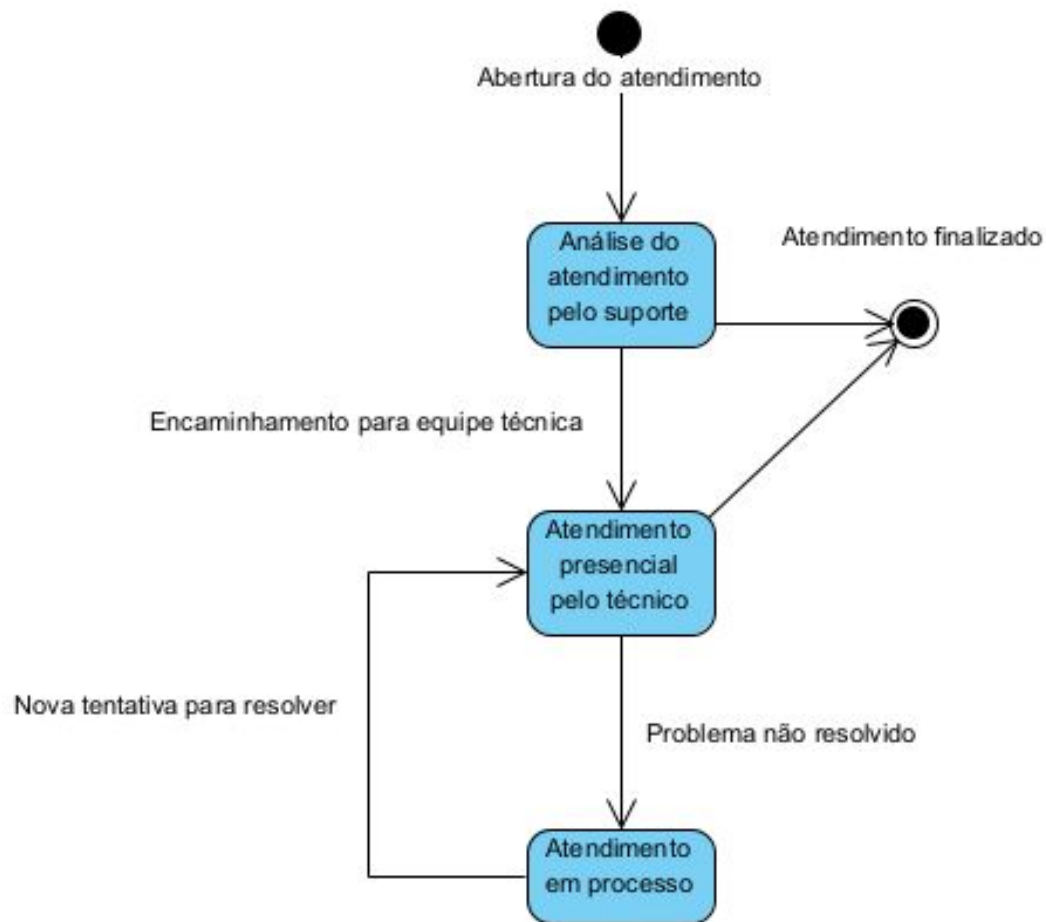
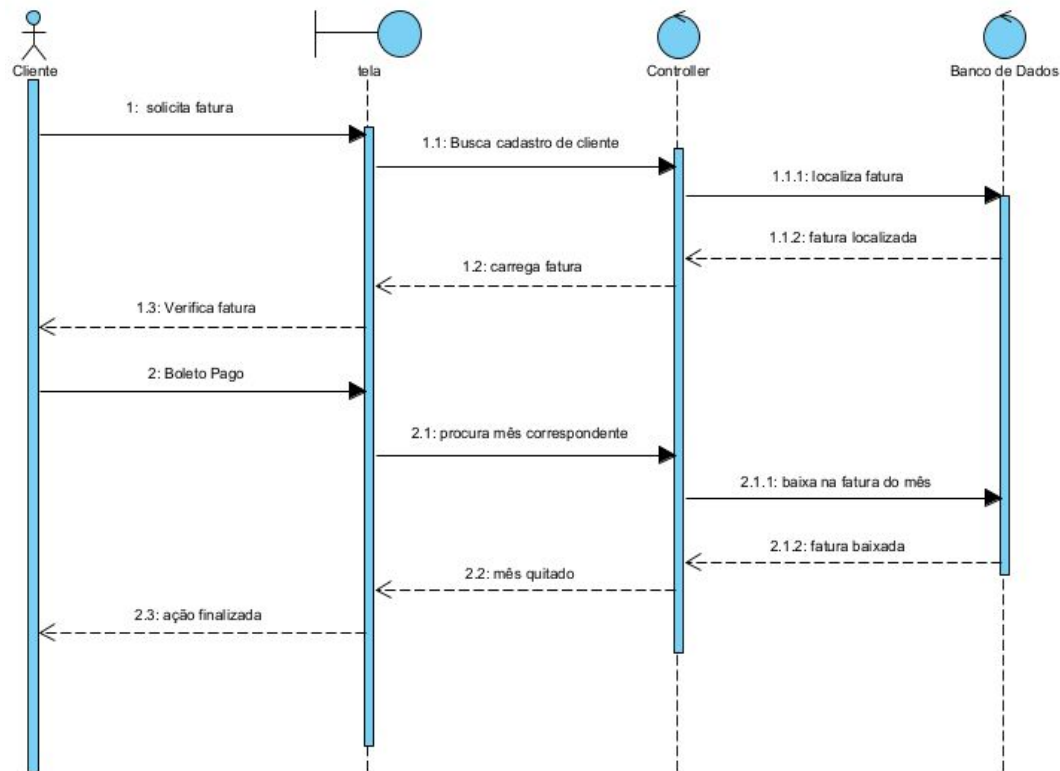


Diagrama de sequência



PROGRAMAÇÃO II

Construção da classe application

```
/**Classe para rodar a aplicacao.
 * @author InsideProvider
 * @version 1.00
 * @since Release da aplicação
 */

@EntityScan(basePackages = "br.edu.unoesc.springboot.insideprovider.model")
@SpringBootApplication
public class Application {

    /** Metodo Main da aplicacao
     * @param args - inicia aplicacao
     */
    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

Classe Model

```
/**Classe para objetos do tipo Usuario, onde serão contidos, valores e métodos para o mesmo.
 * @author InsideProvider
 * @version 1.00
 * @since Release da aplicação
 */

@Entity
@Table(name="usuario")
public class Usuario implements Serializable{
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    @Column(name="cod_usu")
    private Long codUsu;

    @Column(name="cpf_cnpj_usu")
    private Double cpfCnpjUsuario;

    @Column(name="nom_usu")
    private String nomUsu;

    @Column(name="dat_nas_usu")
    private Date nascimentoUsuario;
```



Criação da interface UsuarioRepository

```
/**Interface para Repository de Usuario.  
 * @author InsideProvider  
 * @version 1.00  
 * @since Release da aplicação  
 */  
  
@Repository  
public interface UsuarioRepository extends JpaRepository<Usuario, Long> {  
  
}
```


Classe Service



- Realização de CRUD

```
/**Interface para Service de Usuario.  
* @author InsideProvider  
* @version 1.00  
* @since Release da aplicação  
*/  
  
public interface UsuarioService {  
  
    List<Usuario> findAllUsuario();  
    Optional<Usuario> findById(Long codUsu);  
    Usuario saveUsuario(Usuario usuario);  
    Usuario updateUsuario(Usuario usuario);  
    void deleteUsuario(Long codUsu);  
}
```

Classe Service Implementation



```
/**Classe para Implementar o Service de Usuario.
 * @author InsideProvider
 * @version 1.00
 * @since Release da aplicação
 */

@Service
public class UsuarioServiceImpl implements UsuarioService {

    private final UsuarioRepository usuarioRepository;

    public UsuarioServiceImpl(UsuarioRepository usuarioRepository) {
        this.usuarioRepository = usuarioRepository;
    }

    /** Metodo para listar usuarios
     * @return List<Usuario>
     */
    @Override
    public List<Usuario> findAllUsuario() {
        // TODO Auto-generated method stub
        return usuarioRepository.findAll();
    }
}
```

Classes Controller:



```
@RestController
@RequestMapping("/usuario")
public class UsuarioController {

    private final UsuarioService usuarioService;

    public UsuarioController(UsuarioService usuarioService) {
        this.usuarioService = usuarioService;
    }

    /** Mapeamento GET para listar usuarios
     * @return List<Usuario>
     */
    @GetMapping
    public List<Usuario> findAllUsuario(){
        return usuarioService.findAllUsuario();
    }
}
```