



Rafael da Silva

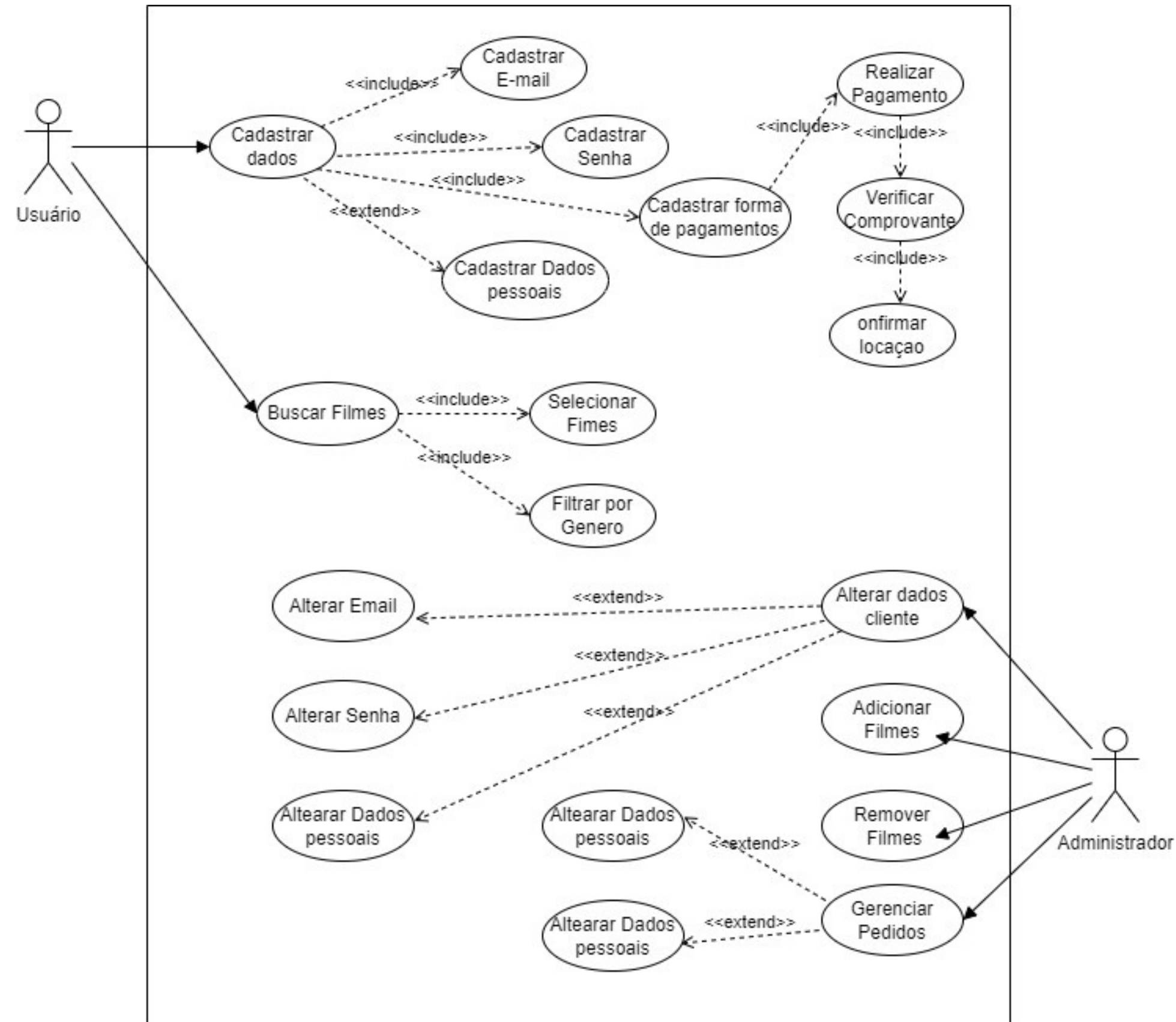
Vicente Sampaio

William Faray

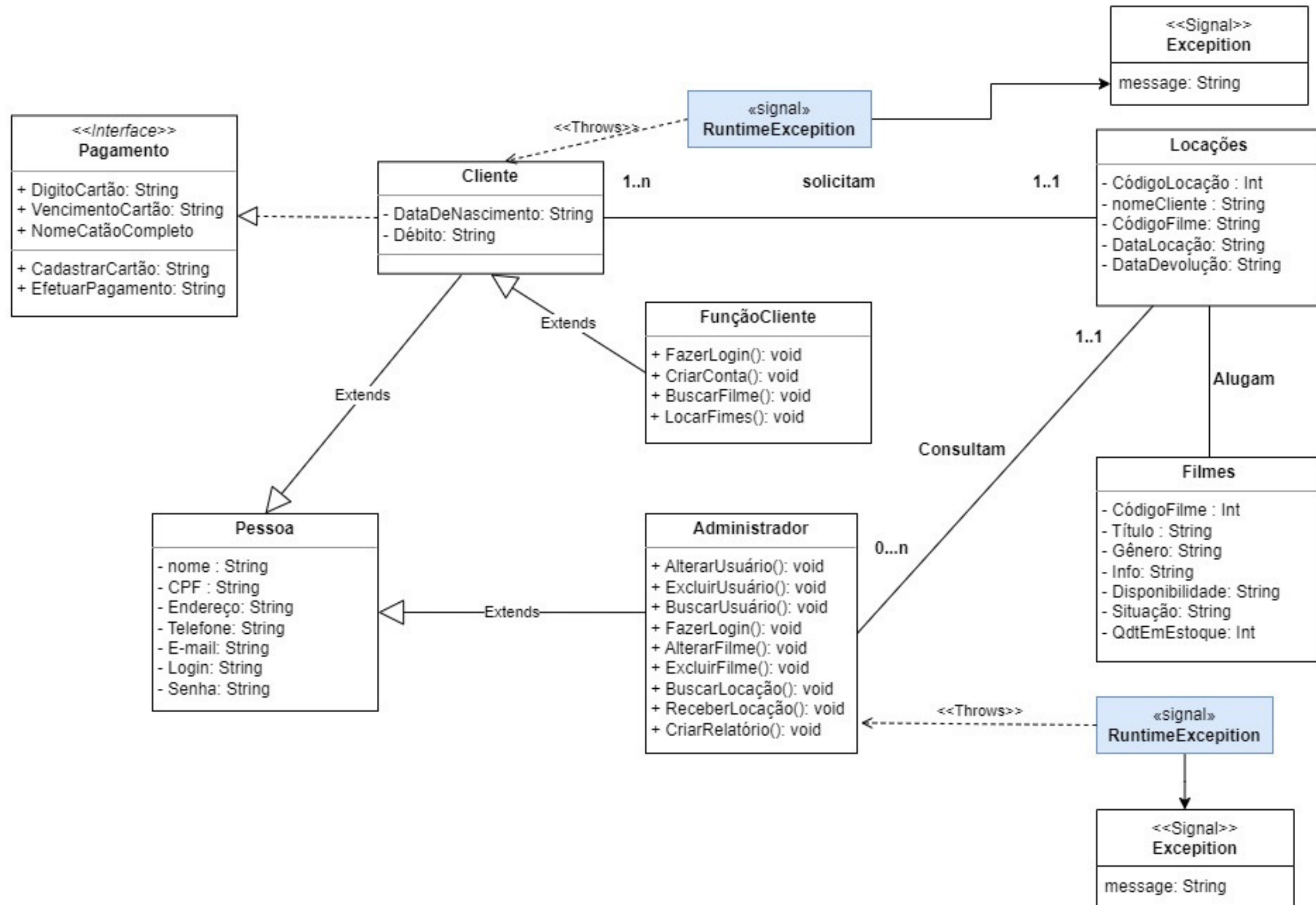
# O que veremos?

- Diagramas do Projeto
- APIs utilizadas
- Herança
- Exception
- Interface
- Polimorfismo
- Banco de Dados
- Aplicação

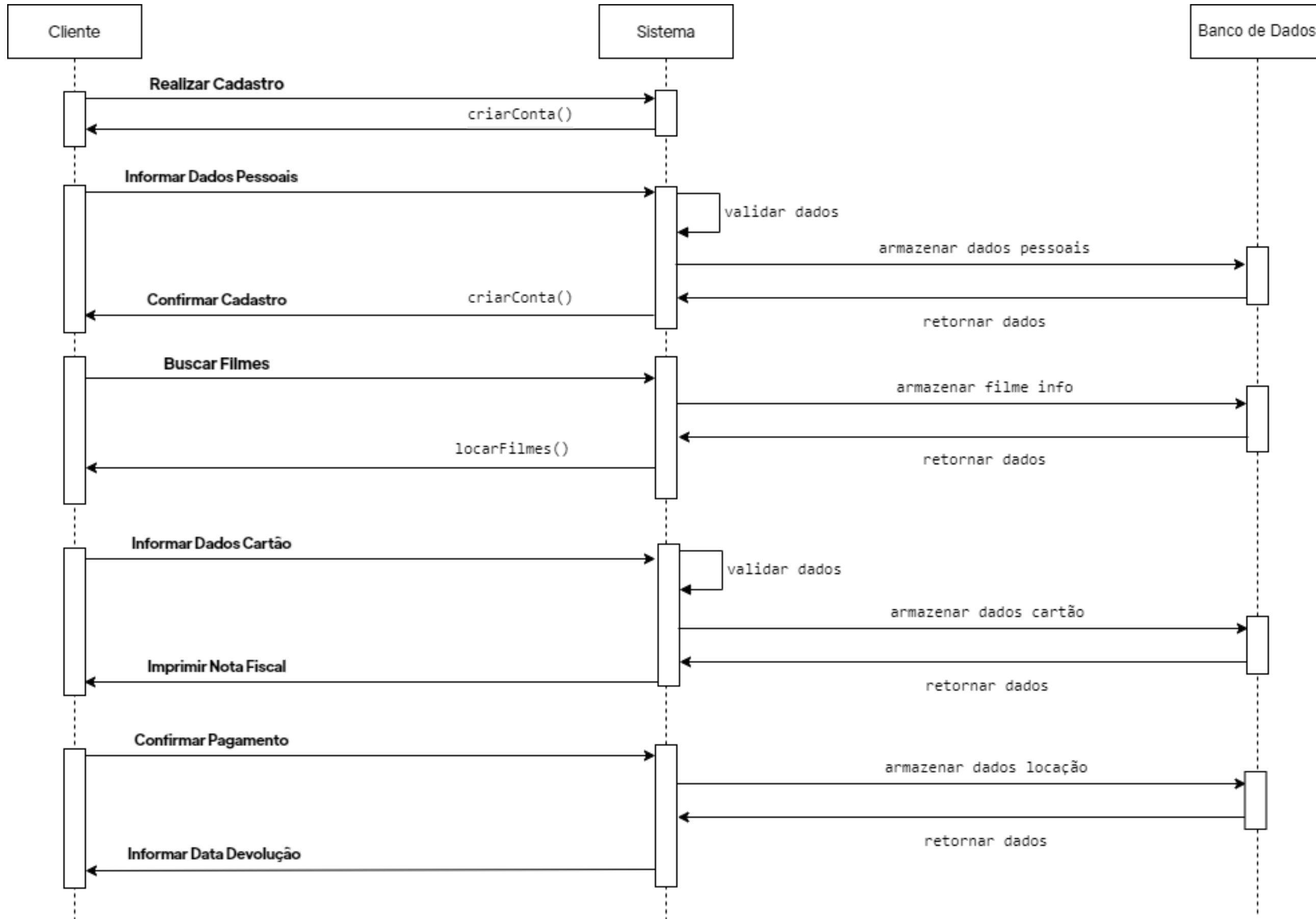
# DIAGRAMA DE CASO DE USO



# DIAGRAMA DE CLASSES



# DIAGRAMA DE SEQUÊNCIA



# APIs UTILIZADAS









































OMDb API



CLOUD TRANSLATION API

- Database aberta de filmes
- Nome, Gênero, Sinopse
- Informações em inglês
- Api de tradução do Google Cloud
- Usada para traduzir informações da OMDb
- Diversos idiomas
- Documentação clara



Nome	Tipo	Esquema
▼  Tabelas (5)		
▼  Carrinho		CREATE TABLE "Carrinho" ( "ID" INTEGER NOT NULL, "Titulo" TEXT NOT NULL, "CodigoFilme" TEXT NOT NULL UNIQUE, "Login" TEXT NOT NULL, PRIMARY KEY("ID" AUTOINCREMENT), FOREIGN KEY("I
 ID	INTEGER	"ID" INTEGER NOT NULL
 Título	TEXT	"Titulo" TEXT NOT NULL
 CodigoFilme	TEXT	"CodigoFilme" TEXT NOT NULL UNIQUE
 Login	TEXT	"Login" TEXT NOT NULL
▼  Desejado		CREATE TABLE "Desejado" ( "ID" INTEGER NOT NULL, "Titulo" TEXT NOT NULL, "CodigoFilme" TEXT NOT NULL UNIQUE, "Login" TEXT NOT NULL, PRIMARY KEY("ID" AUTOINCREMENT) )
 ID	INTEGER	"ID" INTEGER NOT NULL
 Título	TEXT	"Titulo" TEXT NOT NULL
 CodigoFilme	TEXT	"CodigoFilme" TEXT NOT NULL UNIQUE
 Login	TEXT	"Login" TEXT NOT NULL
▼  Locado		CREATE TABLE "Locado" ( "ID" INTEGER, "LoginF" TEXT, "Titulo" TEXT, "CodLocacao" TEXT UNIQUE, "DataLocacao" TEXT, "DataDevolucao" TEXT, "Nome" TEXT, "Mensagem" TEXT, PRIMARY KEY("ID
 ID	INTEGER	"ID" INTEGER
 LoginF	TEXT	"LoginF" TEXT
 Título	TEXT	"Titulo" TEXT
 CodLocacao	TEXT	"CodLocacao" TEXT UNIQUE
 DataLocacao	TEXT	"DataLocacao" TEXT
 DataDevolucao	TEXT	"DataDevolucao" TEXT
 Nome	TEXT	"Nome" TEXT
 Mensagem	TEXT	"Mensagem" TEXT
▼  Pessoa		CREATE TABLE "Pessoa" ( "Nome" TEXT NOT NULL, "CPF" TEXT NOT NULL UNIQUE, "Endereco" TEXT NOT NULL, "Telefone" TEXT NOT NULL, "Email" TEXT NOT NULL UNIQUE, "Login" TEXT NOT NUL
 Nome	TEXT	"Nome" TEXT NOT NULL
 CPF	TEXT	"CPF" TEXT NOT NULL UNIQUE
 Endereco	TEXT	"Endereco" TEXT NOT NULL
 Telefone	TEXT	"Telefone" TEXT NOT NULL
 Email	TEXT	"Email" TEXT NOT NULL UNIQUE
 Login	TEXT	"Login" TEXT NOT NULL UNIQUE
 Senha	TEXT	"Senha" TEXT NOT NULL
 Aniversario	TEXT	"Aniversario" TEXT NOT NULL
 CartaoDigito	TEXT	"CartaoDigito" TEXT UNIQUE
 CartaoMMAA	TEXT	"CartaoMMAA" TEXT
 CartaoNomeCompleto	TEXT	"CartaoNomeCompleto" TEXT UNIQUE
 CartaoRegiao	TEXT	"CartaoRegiao" TEXT
 CartaoCVC	TEXT	"CartaoCVC" TEXT UNIQUE
>  sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
 Índices (0)		
 Vistas (0)		
 Gatilhos (0)		





```
1 package org.Usuario;
2
3 > import ...
4
5
6
7
8
9
10
11 4 usages  📄 Vicente Sampaio *
12 public class Administrador extends Pessoa{
13     2 usages  📄 Vicente Sampaio
14     public Administrador() {
15         try {
16             Class.forName(className: "org.sqlite.JDBC");
17             this.connection = DatabaseProvider.getConnection();
18         } catch (ClassNotFoundException e) {
19             throw new RuntimeException(e);
20         }
21     }
22 }
```

# HERDA DE PESSOA

```
1 package org.Usuario;
2
3 > import ...
13
6 usages 1 inheritor 1 Vicente Sampaio
14 @ public class Cliente extends Pessoa implements SistemaPagamento{
    2 usages
15     private String cartaoDigito;
    2 usages
16     private String cartaoVencimento;
    2 usages
17     private String cartaoNomeCompleto;
    2 usages
18     private String cartaoRegiao;
    2 usages
19     private String cartaoCVC;
20
```

**HERDA DE  
PESSOA**

**IMPLEMENTA  
INTERFACE**

```
1 package org.Sistema;
2
3 2 usages 2 implementations 1 Vicente Sampaio
4 public interface SistemaPagamento {
5     1 usage 1 implementation 1 Vicente Sampaio
6     boolean cadastrarCartao();
7     2 usages 1 implementation 1 Vicente Sampaio
8     void efetuarPagamento();
9
10 }
```

# INTERFACE

```
1 package org.Exception;
2
3 14 usages  📄 Vicente Sampaio
4 public class EntradaInvalidaException extends Exception{
5     11 usages  📄 Vicente Sampaio
6     public EntradaInvalidaException(String mensagem){
7         super(mensagem);
8     }
9 }
```

# EXCEPTION

80

## Classe Pessoa

2 usages 2 overrides  Vicente Sampaio

81  >

```
public boolean fazerLogin() { return false; }
```

84

2 usages 2 overrides  Vicente Sampaio

85  >

```
public boolean criarConta() { return true; }
```

88

2 usages 2 overrides  Vicente Sampaio

89  >

```
public boolean sistemaLogin() { return false; }
```

92

```
}
```

# POLIMORFISMO...

```

26  ~ if (administrador.sistemaLogin()) {
27      Administrador funcoesAdministrador = new Administrador();
28
29  ~ while (true) {
30      System.out.println("""
31          \n|=====|
32          [1] BUSCAR USUÁRIO
33          [2] ALTERAR USUÁRIO
34          [3] EXCLUIR USUÁRIO
35          [4] BUSCAR LOCAÇÃO
36          [5] RECEBER LOCAÇÃO
37          [PRESSIONE ENTER PARA SAIR]
38          |=====|
39          Opção: """);
40
41
42      opcao = ler.nextLine();
43  ~ switch (opcao) {
44          case "1" -> funcoesAdministrador.buscarUsuario();
45          case "2" -> funcoesAdministrador.alterarUsuario();
46          case "3" -> funcoesAdministrador.excluirUsuario();
47          case "4" -> funcoesAdministrador.buscarLocacao();
48          case "5" -> funcoesAdministrador.receberLocacao();
49  ~ default -> {
50              return;
51          }
52      }
53  }
54  }

```

# POLIMORFISMO...

```

55     } else {
56         if (cliente.sistemaLogin()) {
57             while (true) {
58                 System.out.println("""
59                     \n|=====|
60                     [1] BUSCAR FILME
61                     [2] VISITAR SUA LISTA DE DESEJOS
62                     [3] LOCAR FILME
63                     [PRESSIONE ENTER PARA SAIR]
64                     |=====|
65                     Opção: """);
66
67
68                 opcao = ler.nextLine();
69                 switch (opcao) {
70                     case "1" -> cliente.buscarFilme();
71                     case "2" -> cliente.listaDeDesejos();
72                     case "3" -> cliente.locarFilmes();
73                     default -> {
74                         return;
75                     }
76                 }
77             }
78         }

```

**POLIMORFISMO...**



# EQUIPE



## RAFAEL DA SILVA

IDEALIZAÇÃO DO PROJETO

PROGRAMAÇÃO AUXILIAR

DIAGRAMA DE CASO E USO

DIAGRAMA DE CLASSE

TESTES E REPARO DE BUGS

SLIDES

## VICENTE FRANCISCO

IDEALIZAÇÃO DO PROJETO

PROGRAMAÇÃO PRINCIPAL

API OMDB

BANCO DE DADOS

TESTES E REPARO DE BUGS

SLIDES

## WILLIAM FARAY

IDEALIZAÇÃO DO PROJETO

PROGRAMAÇÃO AUXILIAR

API CLOUD TRANSLATION

DIAGRAMA SEQUENCIAL

TESTES E REPARO DE BUGS

SLIDES