deti universidade de aveiro
departamento de eletrónica,
telecomunicações e informática

Sistema de Gestão de Cruzeiros

Bases de Dados

Docentes Joaquim Sousa Pinto e Carlos Costa

Eva Bartolomeu, 98513 Engenharia Informática
Marta Fradique, 98629 Engenharia Informática

Conteúdo

1. Introdução

Como projeto final da cadeira de base de dados tivemos como finalidade desenvolver um sistema de gestão de Cruzeiros.O sistema pode ser utilizado por uma agência de viagens ou pela própria empresa de cruzeiros. Como esta rede por vezes se torna extensa e difícil de gerir, criamos 4 funcionalidades principais:consultar, editar, eliminar e criar novos, e estas são aplicadas aos elementos “barco”, “cruzeiro”, “cliente”, “tripulante” e “reserva”.

Os nossos principais objetivos são utilizar todos os conceitos aprendidos nas aulas, e aplicar estes de forma correta e útil.

2. Ficheiros entregues em anexo:

- SQL DDL.sql (Criação das tabelas da base de dados, aplicação de DDL)
- Inserçoes (Preencher as tabelas com dados)
- Index.sql (Todos os índices utilizados)
- SP.sql (Todos os Stored Procedures criados)
- UDF.sql (Todos as UDF criadas)
- Views.sql (Todos as views criadas)
- Pasta “interface” com a interface criada para a gestão e manipulação da base de dados.
- Video da apresentaçao (vídeo com uma pequena demonstração da interface)

3. Análise de requisitos

O Sistema de gestão de uma agência de cruzeiros tem como objetivo ser capaz de modelar algumas funcionalidades segundo as seguintes premissas:

- A agência contêm vários cais que são caracterizados por localidade, código e nome.
- A agência comercializa vários cruzeiros que são caracterizados por um número, data de embarque e de desembarque, vagas e código de barco.
- Para cada cais existe um codigo o seu nome e o nome da localidade.
- Em cada reserva, temos o nome do responsável da reserva e o seu número de telemóvel;
- A reserva contém vários bilhetes, cada bilhete é caracterizado por um número, o nome do cliente, a data, o número do cruzeiro e número do CC.
- Um barco é caracterizada por um código e por um número total de bilhetes, o nome do barco e o tipo de barco.
- Os tipos de barcos que podem existir é um IATE, Navio Hotel ou normal.
- Os tipos de barcos possuem um nome, e um número máximo de bilhetes que podem suportare a classificação.
- Os tipos de barcos podem atracar em cais.
- Uma Pessoa tem número de telemóvel, nome, email e número CC.
- Pessoa tem tripulantes e clientes.
- Tripulantes tem número de tripulante e número CC.
- Cliente tem número CC e número cliente.

4. Desenho da base de dados

Na figura seguinte podemos observar o diagrama entidade relacionamento, este foi atualizado após uma reunião com o docente e melhoramos os seguintes aspetos:

- Criamos a entidade Pessoa com os atributos(num(PK), email, nome, numTelemovel).
- A entidade Cliente passa a estar ligada à entidade Pessoa por um IS-A.
- Criamos também a entidade tripulante que também está ligada a Pessoa através de um IS-A.

Estas alterações foram bastante úteis pois melhoramos a lógica e o cabimento da nossa base de dados.

4.1 Desenho Entidade Relacional

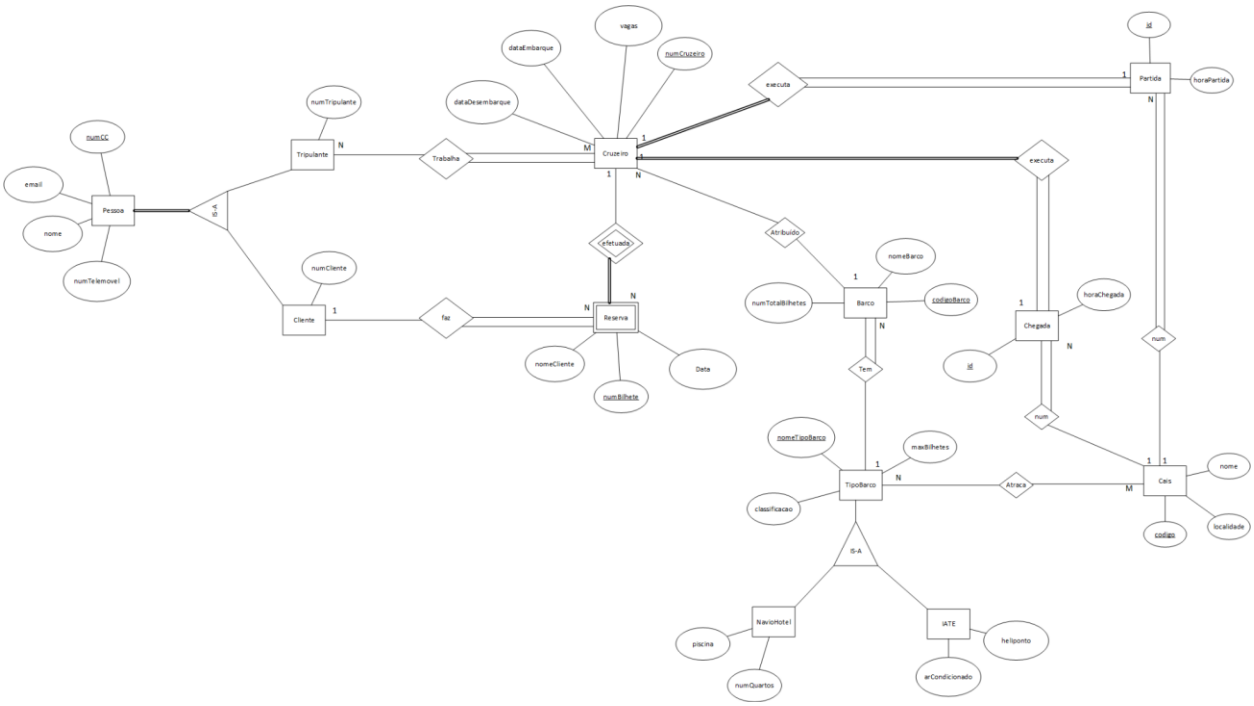

Ilustração 1- Desenho Entidade Relacional

4.2 Esquema Modelo Relacional

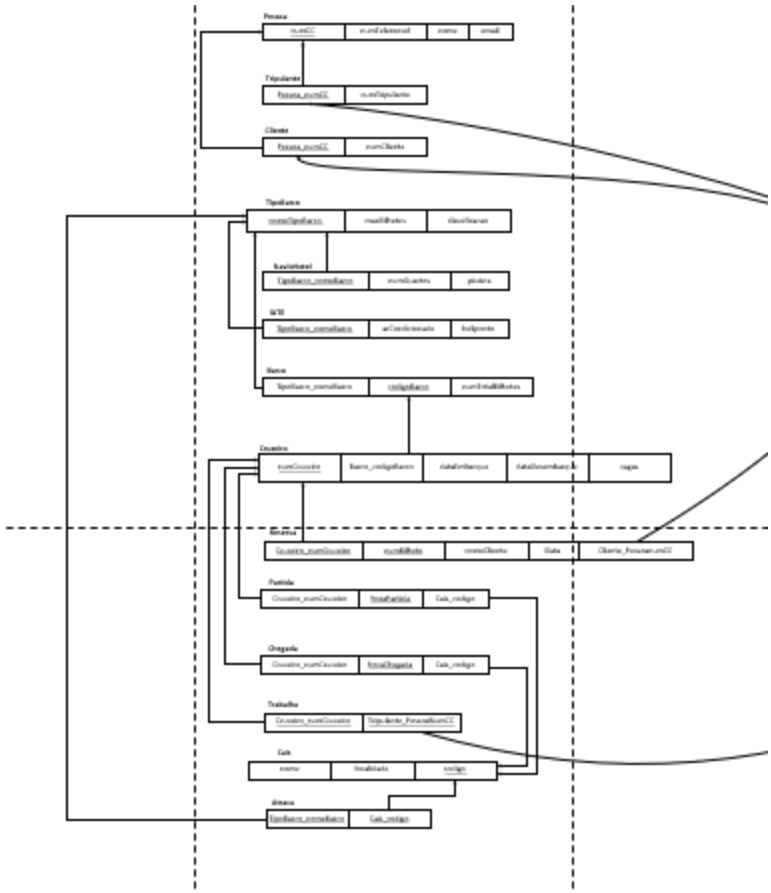

Ilustração 2- Esquema Modelo Relacional

4.3 Modelo Relacional - SQL

De seguida, podemos visualizar o diagrama Modelo Relacional gerado em SQL, após a construção das tabelas.

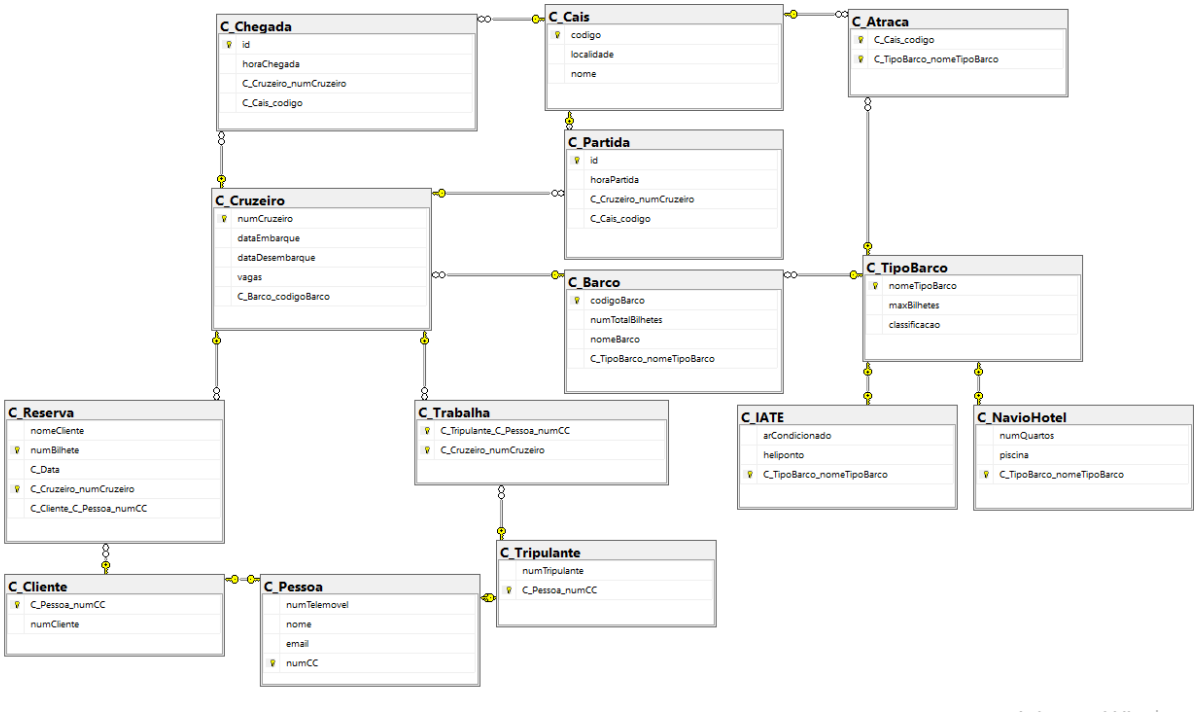

Ilustração 3- Modelo Relacional-SQL

5. Normalização

- Relação Pessoa(numTelemovel, numCC(PK), email, nome):

numCC -> nome
nome -> email, numTelemovel
De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ numTelemovel: A
- ➔ numCC: B
- ➔ email: C
- ➔ nome: D

1FN, 2FN: R = {B, A, C, D}
3FN: R1 = {B, D}
R2 = {D, C, A}

BCNF: R ={B, C, A}

- Relação Tripulante(numTripulante, Pessoa_numCC(PK)(FK)):

Pessoa_numCC -> numTripulante

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ numTripulante: A
- ➔ Pessoa_numCC: B

1FN, 2FN, 3FN, BCNF: R = {B, A}

- Relação Cliente(numCliente, Pessoa_numCC(PK)(FK)):

Pessoa_numCC -> numCliente

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ numCliente: A
- ➔ Pessoa_numCC: B

1FN, 2FN, 3FN, BCNF: R = {B, A}

- Relação TipoBarco(nomeTipoBarco(PK), classificacao, maxBilhetes):

nomeTipoBarco -> maxBilhetes, classificacao

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ maxBilhetes: A
- ➔ nomeTipoBarco: B
- ➔ classificacao: C

1FN, 2FN, 3FN, BCNF: R = {B, A, C}

- Relação NavioHotel(numQuartos, piscina, TipoBarco_nomeBarco(PK)(FK)):

TipoBarco_nomeBarco -> numQuartos, piscina

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ numQuartos: A
- ➔ TipoBarco_nomeBarco: B
- ➔ piscina: C

1FN, 2FN, 3FN, BCNF: R = {B, A, C}

- Relação IATE(arCondicionado, heliponto, TipoBarco_nomeBarco(PK)(FK)):

TipoBarco_nomeBarco -> arCondicionado, heliponto

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ arCondicionado: A
- ➔ TipoBarco_nomeBarco: B
- ➔ heliponto: C

1FN, 2FN, 3FN, BCNF: R = {B, A, C}

- Relação Barco(codigoBarco(PK), numTotalBilhetes, TipoBarco_nomeBarco(FK),nomeBarco):

codigoBarco -> TipoBarco_nomeBarco, nomeBarco

TipoBarco_nomeBarco -> numTotalBilhetes

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ numTotalBilhetes: A
- ➔ codigoBarco: B
- ➔ TipoBarco_nomeBarco: C

1FN, 2FN: R = {B, A, C}
3FN: R1 = {B, C}2
R2 = {C, A}

BCNF: R = {B, A}

- Relação Cruzeiro(numCruzeiro(PK), dataEmbarque, dataDesembarque, vagas, Barco_codigoBarco (FK)):

numCruzeiro -> dataEmbarque, dataDesembarque, Barco_codigoBarco

numCruzeiro, Barco_codigoBarco -> vagas

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ dataEmbarque: A
- ➔ numCruzeiro: B
- ➔ Barco_codigoBarco : C
- ➔ dataDesembarque: D
- ➔ vagas: E

Todas as formas: R = {B, A, C, D, E}

- Relação Reserva(nomeCliente,numBilhete(PK),Data, Cruzeiro_numCruzeiro(FK)(PK),Cliente_PessoanumCC(FK)):

Cliente_PessoanumCC > nomeCliente

Cruzeiro_numCruzeiro, numBilhete -> Data, Cliente_PessoanumCC

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ nomeCliente: A
- ➔ numBilhete: B

- ➔ Cruzeiro_numCruzeiro: C
- ➔ Data: D
- ➔ Cliente_PessoanumCC: E

1FN, 2FN: R = {B, A, C, D, E}

3FN: R1 = {B, C, D, E}

R2 = {E, A}

BCNF: R = {B, C, D,A}

- Relação Cais(codigo(PK),localidade, nome):

codigo -> localidade, nome

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ localidade: A
- ➔ codigo: B
- ➔ nome: C

Todos: R = {B, A, C}

- Relação Partida(id(PK), horaPartida, Cruzeiro_numCruzeiro(FK), Cais_codigo(FK)):

id -> horaPartida, Cais_codigo, Cruzeiro_numCruzeiro

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ Cais_codigo: A
- ➔ id: B
- ➔ Cruzeiro_numCruzeiro: C
- ➔ horaPartida : D

Todas: R = {B, A, C,D}

- Relação Chegada(id(PK), horaChegada, Cruzeiro_numCruzeiro(FK), Cais_codigo(FK)):

id -> horaChegada, Cais_codigo, Cruzeiro_numCruzeiro

De forma a facilitar a decomposição das relações vamos atribuir letras aos atributos:

- ➔ Cais_codigo: A
- ➔ id: B
- ➔ horaChegada: D
- ➔ Cais_codigo: C

Todas: R = {B, A, C,D}

- Relação Trabalha(Tripulante_PessoaNumCC(PK)(FK), Cruzeiro_numCruzeiro(PK)(FK)):

Não tem.

- Relação Atraca(Cais_codigo(PK)(FK), TipoBarco_nomeBarco(PK)(FK)):

Não tem.

6. Construção da Base de Dados

Para construir e manipular a base de dados recorremos à linguagem SQL, particularmente *a Data Definition Language(DDL) e Data Manipulation Language(DML).*

Ainda, criamos um *schema* com o nome *CruzeirosDB.*

6.1. Data Definition Language (DDL)

Para criar as diferentes entidades na base de dados, utilizamos a sintaxe *DDL* e o *SQL Server.* Empregamos estes conhecimentos para criar tabelas e definir os respetivos atributos e o seu tipo de dados. Podemos ver alguns exemplos a seguir:

```
CREATE TABLE CruzeirosDB.C_Partida(
    id int NOT NULL PRIMARY KEY,
    horaPartida time NOT NULL,
    C_Cruzeiro_numCruzeiro int,
    C_Cais_codigo int NOT NULL,
    FOREIGN KEY (C_Cruzeiro_numCruzeiro) REFERENCES CruzeirosDB.C_Cruzeiro(numCruzeiro),
    FOREIGN KEY (C_Cais_codigo) REFERENCES CruzeirosDB.C_Cais(codigo)
)
GO
```

Ilustração 4- Example DDL

6.2. Data Manipulation Language (DML)

Para inserir os dados nas tabelas e fazer consultas simples à tabela utilizamos as instruções de *Data Manipulation Language (DML)*. Ora, também incorporamos estas em *Stored Procedures* e *User Defined Functions*. Alguns exemplos:

```
--Inserir os valores na tabela C_TipoBarco,(12)
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Argos',100,4);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Cranchi',175,3);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Fletcher',170,4);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Johnson',150,5);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Rinker',150,3);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Dufour',200,3);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Karnic',250,5);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Bayliner',190,4);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('FibraMar',150,3);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('SeaDoo',190,4);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Gobbi',100,3);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('RiaMar',110,5);
INSERT INTO CruzeirosDB.C_TipoBarco([nomeTipoBarco],[maxBilhetes],[classificacao]) VALUES('Suzuki',180,4);
```

Ilustração 5- Example DML

7. Interface Gráfica

Para criar a interface gráfica utilizamos o *Visual Studio* e aplicamos um *Windows Forms* da plataforma *.NET*, ainda, utilizamos o *C#* para desenvolver o código.

Como pretendíamos que a nossa interface fosse algo acessível de utilizar e útil, criamos quatro funcionalidades principais: criar, consultar, editar e eliminar os elementos(barcos, cruzeiros, clientes, pessoas e reservas). Para que estas funcionalidades fossem possíveis recorremos a comandos em *SQL-DML(INSERT, DELETE e UPDATE*), *Views, Index, UDF, Stored Procedure* e *Trigger*.

7.1 Cruzeiros

Para criar o cruzeiro Implementamos um *stored procedure(CruzeirosDB.criarCruzeiro),* criamos um *trigger(CruzeirosDB.TriggerAddBarcoTipoBarco)* que valida o tipo de barco e também criamos outro *trigger(CruzeirosDB.TriggerAddCruzeiroData)* que não permite que a data de embarque seja posterior à de desembarque.

Para editar o cruzeiro utilizamos *stored procedure(CruzeirosDB.editCruzeiro)* e para apagar o cruzeiro usamos o *stored procedure(CruzeirosDB.DeleteCruzeiro)*, ainda, criamos também duas *UDF(CruzeirosDB.gethorapartida, CruzeirosDB.gethorachegada, CruzeirosDB.getIdPartida e CruzeirosDB.getIdChegada)* para obter a hora e o ID da partida e da chegada, e tal como na criação também fizemos uso do *trigger(CruzeirosDB.TriggerAddBarcoTipoBarco)* que valida o tipo de barco.

Para fazer uma listagem dos cruzeiros e dos seus atributos utilizamos uma *view(CruzeirosDB.listCruzeiro).*

7.2 Barcos

Para criar a lista com todos os barcos e os seus repetitivos atributos utilizamos um *SELECT.*

Na opção de editar os atributos de barcos já existentes utilizamos um *UPDATE,* para o utilizador eliminar um barco utilizamos um update e criamos um *stored procedure(CruzeirosDB.DeleteBarco)* com um *cursor* que ao eliminarmos um barco elimina também todos os cruzeiros associados a este, e ao eliminar um cruzeiro elimina também todas as partidas e chegadas associadas.

Ao criar um novo barco utilizamos um *trigger(CruzeirosDB.triggerAddBraco)* que não permite que o número total de bilhetes seja superior ao número máximo de bilhetes do tipo de barco que este barco pertence, também criamos um *trigger(CruzeirosDB.TriggerAddBarcoTipoBarco)* que apenas deixa o utilizador atribuir um tipo de barco ao barco que está a criar.

7.3 Cliente eTripulante

A interface de cliente e tripulante são bastante semelhantes por ambas as entidades derivarem de Pessoa. Para implementar ambas as interfaces, utilizamos uma *Stored Procedure(CruzeirosDB.Cliente, CruzeirosDB.Tripulante)* para listar os clientes e tripulantes que já se encontram inseridos na base de dados, e utilizamos *stored procedures(CruzeirosDB.AddCliente, CruzeirosDB.AddTripulante)* com *transaction* para criar clientes e tripulantes novos.

7.4 Reservas

Ao construir a reserva além das outras opções que não diferem das anteriores, implementamos uma barra de pesquisa onde podemos procurar por uma reserva pelo nome do cliente. Para implementar esta funcionalidade de forma a que a pesquisa seja feita de forma rápida, utilizamos um *index(reserva_idx).* Também utilizamos um *trigger(CruzeirosDB.triggercheckVagas)* que verifica se ainda existem vagas disponíveis no cruzeiro em questão.

8. Ferramentas Utilizadas

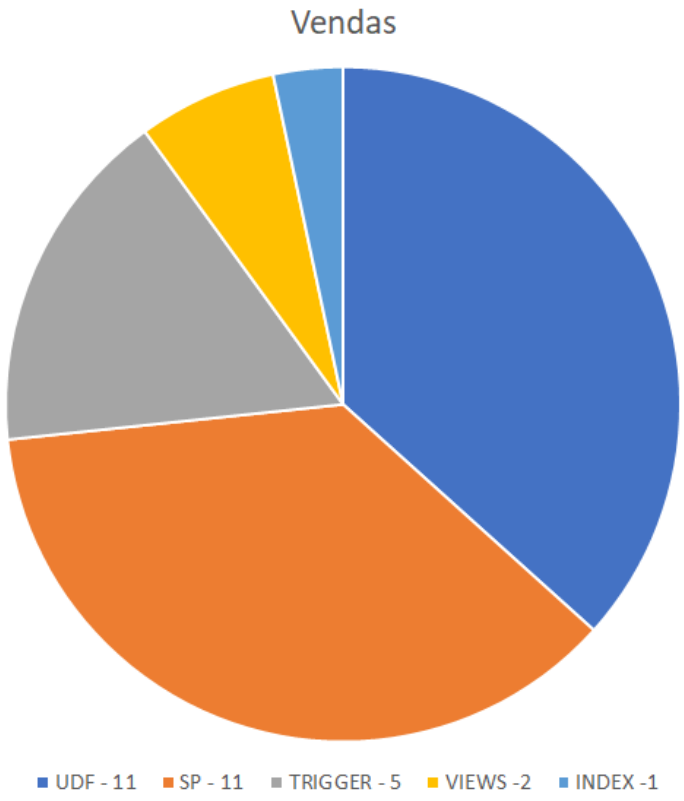

Ilustração 6- ferramentas utilizadas

9. Conclusão

Em conclusão os objetivos iniciais foram cumpridos, visto que aplicamos os conceitos teóricos aprendidos nas aulas. Tivemos algumas dificuldades em trabalhar com o visual code e com C# por não serem ferramentas que estamos habituadas a utilizar.

Como melhoria gostaríamos de ter aplicado conceitos de segurança ao nosso projeto.