

|  |
| --- |
| Complementos de Bases de Dados 2022/2023  Licenciatura em Engª. Informática |
| 1ª Fase Relatório Técnico |

Turma: 2ªL\_EI-SW-08

Horário de Laboratório: 12:30h – 16:30h

Docente: Gabriel Pestana

Grupo

Nº202000753, Nuno Reis

# Introdução

Este relatório tem como objetivo documentar e apresentar algumas justificações para as principais decisões tomadas na primeira fase de desenvolvimento do projeto de CBD, projeto esse que visa a familiarização com a administração de bases de dados relacionais. Ao realizar este projeto será consolidada e posta em prática toda a matéria teórica.

A Wide World Importers (WWI), é uma empresa importadora e distribuidora de produtos que opera no mercado de vendas a retalho. O seu sistema de informação está baseado numa base de dados não normalizada e está desatualizado. A administração tomou a decisão de reformular o sistema, num novo ERP que lhe permita gerir, de forma integrada, todo o processo de vendas.

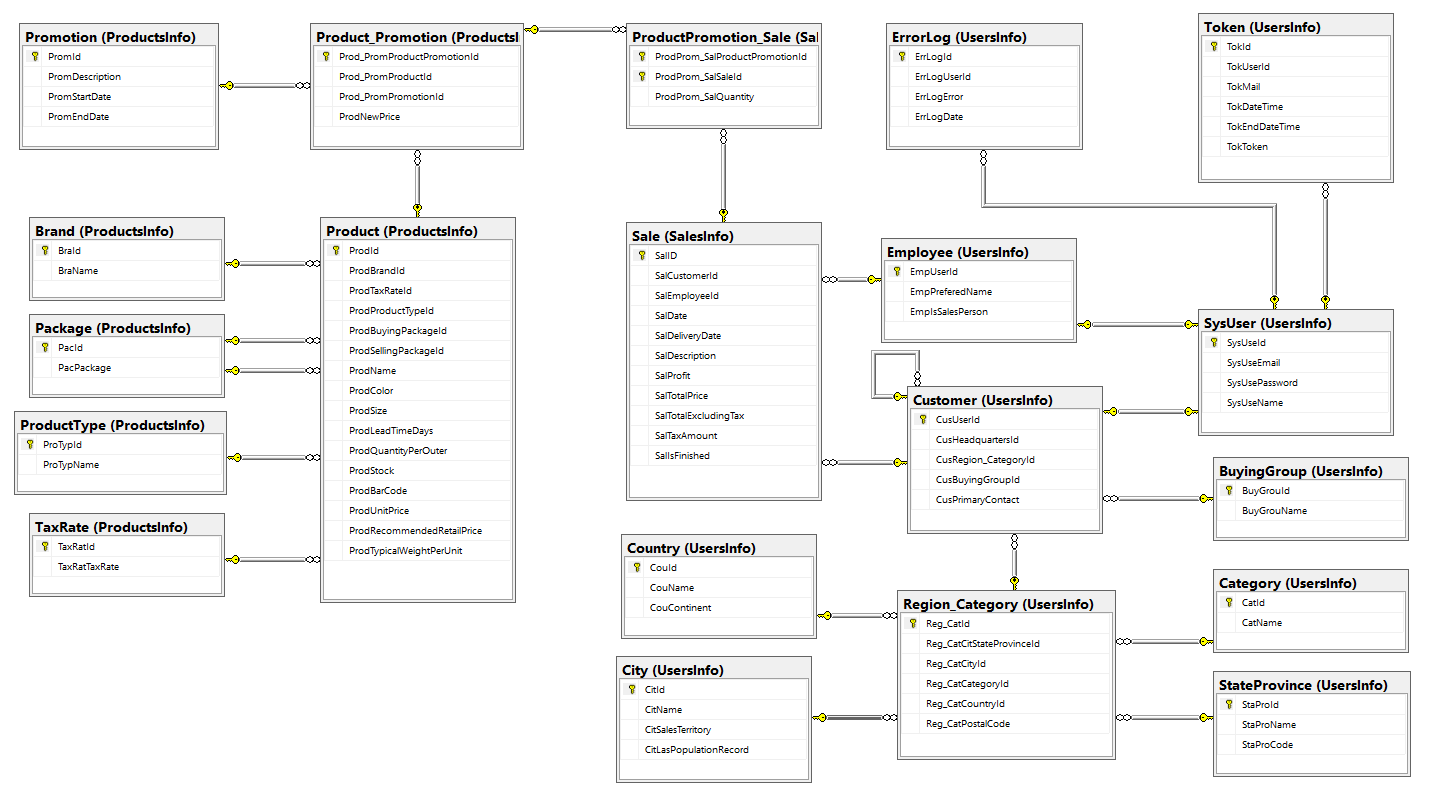
O projeto irá incidir na modelação e integração de uma nova base de dados, que dará apoio a esse novo sistema de informação. Foram exportados e disponibilizados fragmentos de informação do sistema existente, estes apresentam-se fracamente relacionados e carecem de uma otimização segundo as boas praticas de modelação e regras da normalização.

# Especificação de Requisitos

|  |  |  |
| --- | --- | --- |
| ID | Descrição | Implementado  (S/N) |
| RF01 | O sistema não deverá permitir importar o mesmo registo duas vezes. | S |
| RF02 | O sistema deverá guardar o registo de tokens gerados (tokens gerados ao tentar recuperar password). | S |
| RF03 | O sistema deverá verificar se um token está ativo (tokens têm validade de 24 horas) | S |
| RF04 | O sistema não deverá permitir que numa venda existam produtos que necessitem de refrigeração e produtos que não necessitem. | S |
| RF05 | O sistema não deverá permitir a atribuição de uma promoção que não esteja ativa. | S |
| RF06 | O sistema deverá eliminar uma venda se forem removidos todos os produtos. | S |
| RM01 | O sistema deverá guardar a password do utilizador usando codificação uma em SHA1. | S |
| RM02 | O sistema deverá alterar a validade de um token quando um utilizador criar um segundo token. De forma a existir apenas um token valido para cada utilizador. | S |
| RM03 | O sistema deverá calcular a data prevista de entrega e os valores associados a uma venda, assim que esta é finalizada. | S |
| RM04 | O sistema deverá alterar o stock de um produto quando este é adicionado ou removido de uma venda. | S |
| RM05 | O sistema não deverá permitir que existam dois utilizadores com o mesmo email. | S |
| RM06 | O sistema não deverá permitir adicionar um produto a uma venda acabada. | S |

# Modelo Relacional (*Modelo de dados*)

## Diagrama do Modelo Relacional



# Definição do Layout

## Identificação do espaço ocupado por tabela

|  |  |  |
| --- | --- | --- |
| Nome Tabela | Dimensão do Registo | Nº de Registos (inicial/final) |
| UsersInfo.Country | (4+20+20) **44** bytes por cada registo, apenas de **1** registo por ano, ou seja, **44** bytes por ano. | Inicial:  Final: |
| UsersInfo.City | (4+50+50+4) **108** bytes por cada registo, mais de **23272** registos por ano, ou seja, **2513376** bytes por ano. | Inicial:  Final: |
| UsersInfo.StateProvince | (1+50+5) **56** bytes por cada registo, mais de **60** registos por ano, ou seja, **3360** bytes por ano. | Inicial:  Final: |
| UsersInfo.Category | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.BuyingGroup | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.SysUser | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.Region\_Category | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.Customer | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.Employee | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.Package | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.Brand | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.ProductType | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.TaxRate | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.Product | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.Promotion | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| ProductsInfo.Product\_Promotion | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| SalesInfo.Sale | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| SalesInfo.ProductPromotion\_Sale | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.ErrorLog | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |
| UsersInfo.Token | (1+2+3+4+4) **t** bytes por cada registo, mais de **n** registos por ano, ou seja, **t\*n** bytes por ano. | Inicial:  Final: |

## Especificação dos FIlegroups

|  |  |  |
| --- | --- | --- |
| Nome Filegroup | Tabelas associadas | Parâmetros |
| UsersInfo | Country  City  StateProvince  Category  BuyingGroup  SysUser  Region\_Category  Customer  Employee  ErrorLog  Token | Dimensão inicial:  Dimensão final:  Taxa de crescimento: |
| ProductsInfo | TaxRate  Promotion  ProductType  Product\_Promotion  Product  Package  Brand | Dimensão inicial:  Dimensão final:  Taxa de crescimento: |
| SalesInfo | Sale  ProductPromotion\_Sale | Dimensão inicial:  Dimensão final:  Taxa de crescimento: |

## Schemas

|  |  |
| --- | --- |
| Nome | Descrição |
| UsersInfo | Este schema tem como objetivo o agrupamento das tabelas que contêm dados relacionados com os utilizadores. |
| ProductsInfo | Este schema tem como objetivo o agrupamento das tabelas que contêm dados relacionados com os produtos e promoções. |
| SalesInfo | Este schema tem como objetivo o agrupamento das tabelas que contêm dados relacionados com as vendas. |
| OldData | Este schema tem como objetivo o suporte á importação dos dados, primeiro os dados são importados diretamente para este schema sem nenhum tratamento e posteriormente são importados para as tabelas correspondentes. |

# Verificação da migração de dados

## Consultas sobre a base de dados original

No final do ficheiro *WWWI\_DS Query.sql* estão presentes as seguintes consultas:

* Número de clientes
* Número de clientes por categoria
* Total de vendas por funcionário
* Total monetário de vendas por produto
* Total monetário de vendas por produto por ano
* Total monetário de vendas por cidade por ano

## Consultas sobre a nova base de dados

No final do ficheiro *WWWIGlobal Query.sql* estão presentes as seguintes consultas:

* Número de clientes
* Número de clientes por categoria
* Total de vendas por funcionário
* Total monetário de vendas por produto
* Total monetário de vendas por produto por ano
* Total monetário de vendas por cidade por ano

# Programação

## Views

|  |  |
| --- | --- |
| Nome | Descrição |
| UsersInfo.viewNorthAmericaCountry | Esta view permite obter a lista de países na américa do norte. |
| UsersInfo.viewCitySalesTerritory | Esta view permite obter a lista de território de vendas. |
| UsersInfo.viewRegion\_Category | Esta view permite obter a lista de relações entre cidade, estado, continente e categoria, com os campos das chaves externas. |
| UsersInfo.viewCustomer | Esta view permite obter a lista de customers, com os campos das chaves externas. |
| UsersInfo.viewEmployee | Esta view permite obter a lista funcionários. |
| ProductsInfo.viewProduct | Esta view permite obter a lista de produtos, com os campos das chaves externas. |
| ProductsInfo.viewProductPromotion | Esta view permite obter a lista de produtos associados a promoções, com os campos das chaves externas. |
| SalesInfo.viewProductPromotionSale | Esta view permite obter a lista de produtos e promoções associados a vendas, com os campos das chaves externas. |

## Functions

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Atributos | Requisito | Descrição |
| UsersInfo.countryExists | @country varchar(30) | RF01 | Caso exista um país com o nome passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.stateExistsByName | @StateProvince varchar(50) | RF01 | Caso exista um estado com o nome passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.stateExistsByCode | @StateProvince varchar(30) | RF01 | Caso exista um estado com o código passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.cityExists | @City varchar(50) | RF01 | Caso exista uma cidade com o nome passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.categoryExists | @Category varchar(30) | RF01 | Caso exista uma categoria com o nome passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.buyingGroupExists | @buyingGroup varchar(30) | RF01 | Caso exista uma empresa com o nome passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.fnHashPassword | @password VARCHAR(20) | RM01 | Encripta a password passada usando codificação uma em SHA1. |
| UsersInfo.userExistsByName | @name varchar(50) | RF01 | Caso exista um utilizador com o nome passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.Region\_CategoryExists | @State varchar(30)  @City varchar(30)  @Category varchar(30) | RF01 | Caso exista uma relação entre um estado, uma cidade e uma categoria com os nomes passados retorna o seu id caso contrário retorna 0. |
| UsersInfo.CustomerExists | @UserID int | RF01 | Caso exista um cliente com o nome id retorna o seu id caso contrário retorna 0. |
| UsersInfo.EmployeeExists | @UserID int | RF01 | Caso exista um funcionário com o id passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.packageExists | @name varchar(50) | RF01 | Caso exista um pacote com o nome passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.brandExists | @name varchar(50) | RF01 | Caso exista uma marca com o nome passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.productTypeExists | @name varchar(50) | RF01 | Caso exista um tipo de produto com o nome passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.taxRateExists | @tax float | RF01 | Caso exista uma taxa com o valor passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.productExists | @name varchar(100) | RF01 | Caso exista um produto com o nome passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.promotionExists | @id int | RF01 | Caso exista uma promoção com o id passado retorna o seu id caso contrário retorna 0. |
| ProductsInfo.productPromotionExists | @name varchar(100)  @id int | RF01 | Caso exista uma relação entre um produto e uma promoção com o nome do produto e o id da promoção passados retorna o seu id caso contrário retorna 0. |
| SalesInfo.vendaExists | @saleID int | RF01 | Caso exista uma venda com o id passado retorna o seu id caso contrário retorna 0. |
| SalesInfo.productPromotion\_VendaExists | @saleID int  @productPromotionID int | RF01 | Caso exista uma relação entre uma venda e a relação entre um produto e uma promoção com os ids passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.userExistsByEmail | @email varchar(50) | RM05 | Caso exista um utilizador com o email passado retorna o seu id caso contrário retorna 0. |
| UsersInfo.userHasErrors | @id int |  | Caso existam erros associados a um utilizador passado retorna o número de erros caso contrário retorna 0. |
| UsersInfo.tokenExists | @userID int, @token int | RF03 | Caso exista um token valido associado a um utilizador passados retorna o seu id caso contrário retorna 0. |
| productPromotionExistsByName | @name varchar(100) @promotion varchar(100) |  | Caso exista uma relação entre um produto e uma promoção com o nome do produto e a descrição da promoção passados retorna o seu id caso contrário retorna 0. |
| SalesInfo.saleExistsByDescription | @description varchar(100) |  | Caso exista uma venda com a descrição passada retorna o seu id caso contrário retorna 0. |
| ProductsInfo.productStock | @product varchar(100) |  | Caso exista um produto com o nome passado retorna o seu stock caso contrário retorna 0. |
| SalesInfo.saleType | @sale varchar(100) |  | Caso exista uma venda com a descrição passada retorna o tipo de produtos que podem ser adicionados caso contrário retorna ‘ ‘. |

## Stored procedures

|  |  |  |  |
| --- | --- | --- | --- |
| Nome | Atributos | Requisito | Descrição |
| dbo.Migrate\_OldData\_CityTable |  | RF01 | Permite migrar os registos da tabela OldData.City para as tabelas UsersInfo.Country, UsersInfo.StateProvince e UsersInfo.City |
| dbo.Migrate\_OldData\_StatesTable |  | RF01 | Permite migrar os registos da tabela OldData.States para a tabela UsersInfo.StateProvince |
| dbo.Migrate\_OldData\_lookupTable |  | RF01 | Permite migrar os registos da tabela OldData.lookup para a tabela UsersInfo.Category |
| dbo.Migrate\_OldData\_CustomerTable |  | RF01 | Permite migrar os registos da tabela OldData.Customer para as tabelas UsersInfo.SysUser, UsersInfo.Customer, UsersInfo.Region\_Category e UsersInfo.BuyingGroup |
| dbo.Migrate\_OldData\_EmployeeTable |  | RF01 | Permite migrar os registos da tabela OldData.Employee para as tabelas UsersInfo.Employee e UsersInfo.Region\_Category |
| dbo.Migrate\_OldData\_ItemStockTable |  | RF01 | Permite migrar os registos da tabela OldData.[Stock Item] para as tabelas ProductsInfo.Package, ProductsInfo. Brand, ProductsInfo.TaxRate, ProductsInfoProductType, ProductsInfo.Product e ProductsInfo.Product\_Promotion |
| dbo.Migrate\_OldData\_SaleTable |  | RF01 | Permite migrar os registos da tabela OldData.Sale para as tabelas SalesInfo.Sale e SalesInfo.ProductPromotion\_Sale |
| dbo.MigrateAll |  | RF01 | Permite executar todos os stored procedures de migração |
| UsersInfo.editSysUser | @userEmail VARCHAR(50)  @newUserName VARCHAR(50)  @newUserEmail VARCHAR(50)  @newUserPassword VARCHAR(20) |  | Permite editar um utilizador. |
| UsersInfo.createSysUser | @userName VARCHAR(50)  @userEmail VARCHAR(50)  @userPassword VARCHAR(20) |  | Permite adicionar um utilizador. |
| UsersInfo.deleteSysUser | @userEmail VARCHAR(50) |  | Permite apagar um utilizador. |
| UsersInfo.createToken | @userEmail VARCHAR(50) |  | Permite criar um token. |
| UsersInfo.recuperarPassword | @userEmail VARCHAR(50)  @token int  @newUserPassword VARCHAR(20) |  | Permite recuperar password. |
| cangePromotionDatesStart | @product VARCHAR(100)  @promotion VARCHAR(100)  @startDate date |  | Permite alterar a data de início de uma promoção. |
| cangePromotionDatesEnd | @product VARCHAR(100)  @promotion VARCHAR(100)  @endDate date |  | Permite alterar a data de fim de uma promoção. |
| cangePromotionDates | @product VARCHAR(100)  @promotion VARCHAR(100)  @startDate date  @endDate date  AS |  | Permite alterar a data de início e fim de uma promoção. |
| SalesInfo.createSale | @customer VARCHAR(100)  @employee VARCHAR(100)  @description VARCHAR(100) |  | Permite criar uma venda. |
| SalesInfo.addProductToSale | @product VARCHAR(100)  @promotion VARCHAR(100)  @description VARCHAR(100)  @quantity int |  | Permite adicionar um produto a uma venda. |
| SalesInfo.changeProductQuantitySale | @product VARCHAR(100)  @promotion VARCHAR(100)  @description VARCHAR(100)  @quantity int |  | Permite alterar a quantidade de um produto numa venda. |
| SalesInfo.removeProductSale | @product VARCHAR(100)  @promotion VARCHAR(100)  @description VARCHAR(100)  @quantity int |  | Permite remover um produto de uma venda. |

## Triggers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nome | Tipo | Tabela | Requisito | Descrição |
| UsersInfo.tr\_validade\_token | INSERT | UsersInfo.Token | RM02 | Altera a data de validade de um token repetido. |
| SalesInfo.tr\_eliminateSale | DELETE | SalesInfo.ProductPromotion\_Sale | RF06 | Elimina uma venda se forem removidos todos os produtos. |
| tr\_calculateSaleInfo | UPDATE | SalesInfo.Sale | RM03 | Calcula a data prevista de entrega e os valores associados a uma venda finalizada |

# Catálogo/Metadados

## Geradores

|  |  |  |
| --- | --- | --- |
| Nome | Atributos | Descrição |
| dbo.geraInsert | @tname VARCHAR(50) … | Implementa o procedimento para inserir registos numa tabela |
|  |  |  |

## Monitorização

|  |  |  |
| --- | --- | --- |
| Nome | Atributos | Descrição |
|  |  |  |

# Descrição da Demonstração

## Script de demonstração

O primeiro passo é criar a base de dados para tal temos de executar o ficheiro *CriacaoBD.sql* (cria a base de dados WWWIGlobal, schemas e tabelas), podemos também executar o ficheiro *Views.sql* para criar as views.

Para a importação dos dados da base de dados antiga para a nova teremos de realizar os seguintes passos:

* Executar todos os ficheiros da pasta Scripts de migração (cria e importa as tabelas e registos da base de dados antiga WWW\_DS), a ordem da execução dos ficheiros apenas importa nos ficheiros Sale…, a ordem tem de ser *Sale1.sql*, *Sale2.sql*, *Sale3.sql* e por fim *Sale4.sql*.
* Para verificar que as tabelas foram bem criadas e os registos foram bem importados podemos executar as queries do ficheiro *WWW\_DS Query.sql* e as primeiras queries do ficheiro *WWWGlobal Query.sql* (apenas as queries associadas ao schema OldData).
* Por fim basta executar o ficheiro *OldDataMigration.sql* para que todos os registos sejam tratados e inseridos nas novas tabelas.
* Podemos também conferir se os registos foram bem tratados e inseridos executando as queries dos ficheiros *WWW\_DS Query.sql* e *WWWGlobal Query.sql*.

Para testar as funcionalidades propostas e desenvolvidas é necessário executar os seguintes ficheiros:

* *Triggers.sql* (cria os triggers).
* *FunctionsStoredProcedures.sql* (cria as funções).
* *Geradores.sql* (cria os stored procedures geradores).

Depois disso basta executar a funcionalidade que pretendemos testar (a forma como se executa todas as funcionalidades está nos ficheiros).

Para gerar e guardar as informações relativas á monitorização e espaço ocupado pelas tabelas e colunas precisamos de executar o ficheiro *monitorização.sql*.

# Conclusões

Com a realização deste projeto fiquei mais familiarizado com a administração de uma base de dados relacional, apesar de não ter conseguido implementar tudo o que era proposto e ambicionei. Consegui aplicar o conhecimento que adquiri em anos anteriores na disciplina Bases de Dados e no ano atual em Complementos de Base de Dados.

Fazer este projeto sozinho foi muito desafiador pois tive de desenvolver todo o projeto apenas com a ajuda do professor e a pesquisa na internet.