Departamento de Eletrónica, Telecomunicações e Informática

Curso 8240 - Mestrado Integrado em Engenharia de Computadores e Telemática

Disciplina 42532 - Base de Dados

Ano letivo 2020/2021

Trabalho Prático Final – Relatório

Autores:
89078 Luís Filipe Correia do Couto – 50% do trabalho
89082 António Jacinto Coelho Ferreira – 50% do trabalho
Turma P1
Grupo 6

Docente Joaquim Manuel Henriques de Sousa Pinto

Resumo Este documento apresenta o relatório do trabalho prático final. É primeiramente apresentada uma análise de requisitos, seguida do Diagrama Entidade-Relacionamento. Após isto, encontram-se tabelas com o esquema relacional e as respetivas chaves de cada relação, seguidas do Diagrama Relacional. Finalmente existe um resumo e uma breve explicação do código SQL concebido para a base de dados e da Interface.

Índice do Relatório

Tema Proposto

Plataforma Digital de Jogos e Aplicações

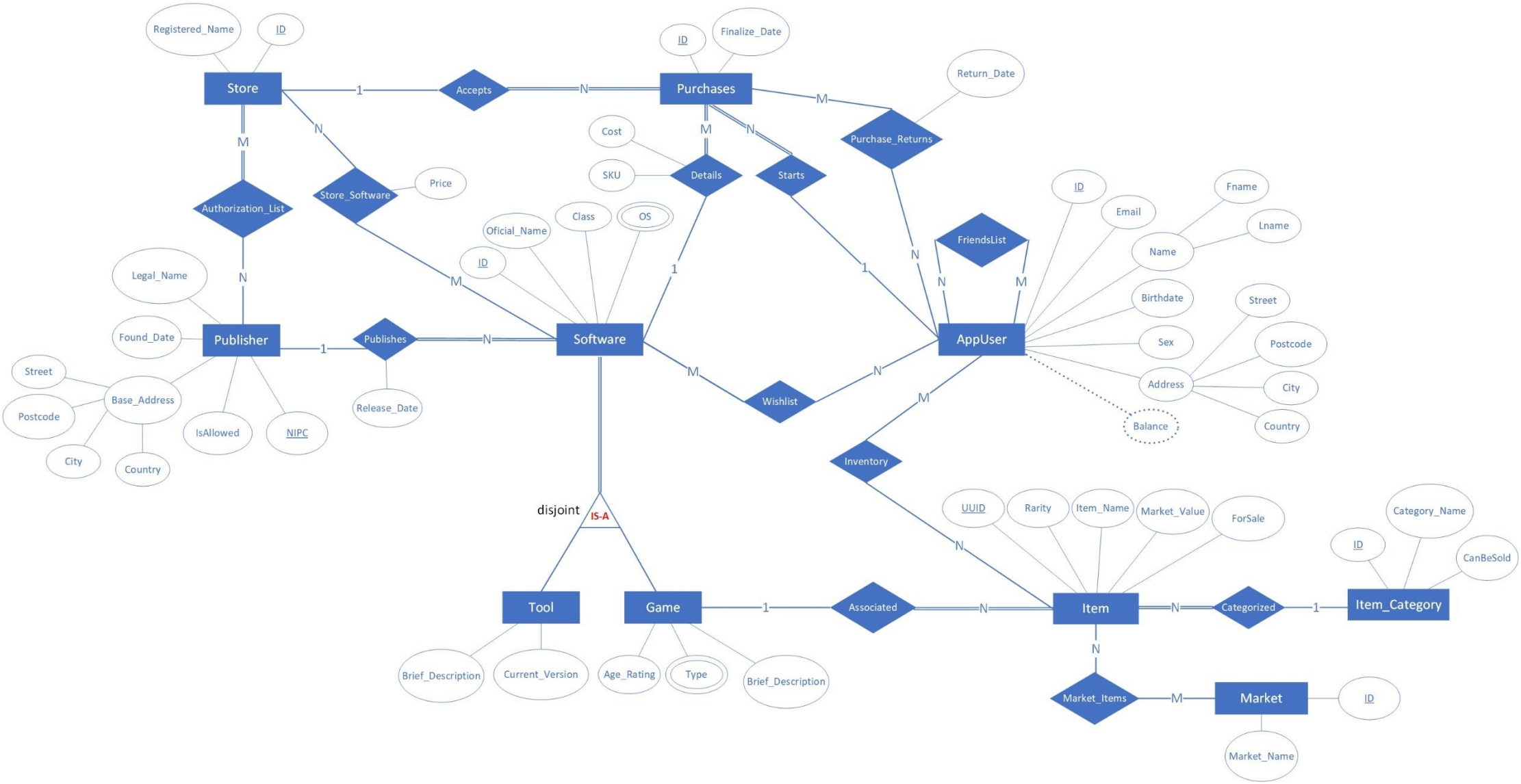
Análise de Requisitos

Este projeto implica implementar uma base de dados para a gestão de uma Plataforma Digital de Jogos e Aplicações. Com isto em mente, e após uma análise em detalhe do sistema, os seguintes requisitos foram determinados:

- É possível gerir a loja, mercado, utilizadores da plataforma, itens, publicadoras e software, assim como ver estatísticas sobre estes.
- Um Utilizador é caracterizado por ID, Email, Nome (Primeiro Nome e Ultimo Nome), Data de Nascimento, Sexo, Morada (Rua, Código Postal, Cidade e País), Servidor e Saldo.
- Uma Editora é caracterizada por NIPC, Nome, Data de Fundação, Endereço (Rua, Código Postal, Cidade e País) e Permissão.
- Uma Transação é caracterizada por ID e Data de Realização.
- Um Item é caracterizado por UUID, Nome, Categoria, Raridade, Valor de Mercado e Indicador de Venda.
- Software é caracterizado por ID, Nome, Subtipo e OS.
- Jogo é um software com Género de Jogo, Restrição de Idade e Descrição.
- Aplicação é um software com Versão e Descrição.
- A loja digital apresenta software associado a um certo preço. Apenas pode ser adicionado na loja software de uma editora autorizada, sendo que existe uma lista de autorização. Podem ser adicionadas/removidas editoras da lista de autorização. Se uma editora for removida da lista, o software desta também é removido da loja.
- Software pode ser removido da loja, mas se já tiver sido adquirido por um utilizador, não pode ser removido da biblioteca deste.
- É possível adicionar saldo a um utilizador. Este saldo é usado para a compra de produtos (software e itens). O utilizador apenas pode comprar um produto se tiver saldo suficiente.
- Uma transação é iniciada por um utilizador e, se for aceite pela loja, é concretizada, ficando associado ao software adquirido um SKU único. Um utilizador pode assim comprar várias instâncias do mesmo software.

- Cada utilizador tem uma biblioteca, que apresenta software já adquirido. Nem aplicações nem jogos podem ser removidos da biblioteca, no entanto podem ser devolvidos, se o tempo de compra não ultrapassar 3 dias, recebendo o utilizador na sua carteira o montante correspondente ao preço do produto.
- Cada utilizador tem uma lista de amigos, de onde podem ser adicionados/removidos outros utilizadores.
- Cada utilizador tem uma lista de desejos, de onde pode ser adicionado/removido software. Software removido da loja é removido da lista de desejos de todos os utilizadores. Software pode ser comprado diretamente a partir desta lista. Quando o software é comprado, é removido desta lista.
- Cada utilizador tem um inventário que contém os seus itens, estando cada um associado a um jogo especifico. Itens de certas categorias podem ser colocados no mercado da comunidade, sendo que se forem comprados, são removidos do inventário do vendedor e colocados no inventario do comprador, assim como o respetivo montante em ambas as carteiras dos utilizadores.
- Um utilizador pode remover do mercado qualquer item que tenha posto à venda, desde que este ainda não tenha sido comprado.
- Quando um item é criado, é automaticamente introduzido no inventario de um utilizador aleatório.
- Se um item for eliminado, o montante correspondente ao seu valor de mercado é adicionado à carteira do utilizador que tinha, no momento da eliminação, o item no seu inventario.
- Quando um utilizador apaga a sua conta, os seus itens também são apagados.

Diagrama Entidade-Relacionamento



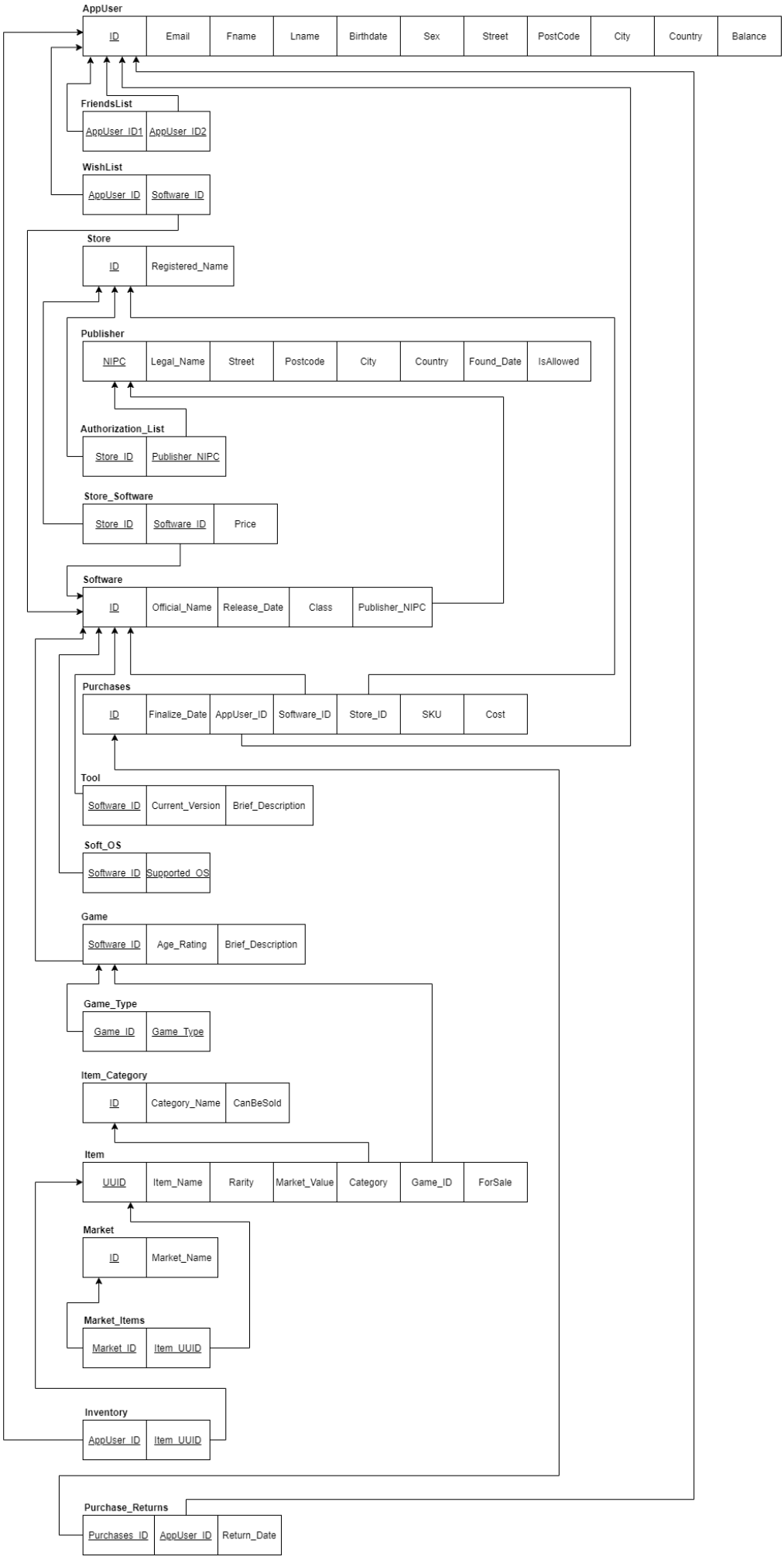
Esquema Relacional

Relação R	Esquema de Relação r(R)
Store	ID, Registered_Name
Market	ID, Market_Name
AppUser	ID, Email, Fname, Lname, Birthdate, Sex, Street, Postcode, City, Country, Balance
Purchases	ID, Finalize_Date, AppUser_ID, Software_ID, Store_ID, SKU, Cost
Publisher	NIPC, Legal_Name, Street, Postcode, City, Country, Found_Date, IsAllowed
Item_Category	ID, Category_Name, CanBeSold
Item	UUID, Item_Name, Rarity, Market_Value, Category, Game_ID, ForSale
Software	ID, Oficial_Name, Release_Date, Class, Publisher_NIPC
Game	Software_ID, Age_Rating, Brief_Description
Tool	Software_ID, Current_Version, Brief_Description
Game_Type	Game_ID, Game_Type
Soft_OS	Software_ID, Supported_OS
Authorization_List	Store_ID, Publisher_NIPC
Store_Software	Store_ID, Software_ID, Price
Market_Items	Market_ID, Item_UUID
Inventory	AppUser_ID, Item_UUID
FriendsList	AppUser_ID1, AppUser_ID2
Wishlist	AppUser_ID, Software_ID
Purchase_Returns	Purchases_ID, AppUser_ID, Return_Date

Chaves das Relações

Relação	Chaves Candidatas	Chave Primária	Chaves Estrangeiras
Store	{ID}	{ID}	-
Market	{ID}	{ID}	-
AppUser	{ID}, {Email}	{ID}	-
Purchases	{ID}, {SKU}	{ID}	{AppUser_ID}, {Software_ID}, {Store_ID}
Publisher	{NIPC}, {Legal_Name}	{NIPC}	-
Item_Category	{ID}, {Category_Name)	{ID}	-
Item	{UUID}	{UUID}	{Category}, {Game_ID}
Software	{ID}	{ID}	{Publisher_NIPC}
Game	{Software_ID}	{Software_ID}	{Software_ID}
Tool	{Software_ID}	{Software_ID}	{Software_ID}
Soft_OS	-	{Software_ID} + {Supported_OS}	{Software_ID}
Game_Type	-	{Game_ID} + {Game_Type}	{Game_ID}
Authorization_List	-	{Store_ID} + {Publisher_NIPC}	{Store_ID}, {Publisher_NIPC}
Store_Software	-	{Store_ID} + {Software_ID}	{Store_ID}, {Software_ID}
Market_Items	-	{Market_ID} + {Item_UUID}	{Market_ID}, {Item_UUID}
Inventory	-	{AppUser_ID} + {Item_UUID}	{AppUser_ID}, {Item_UUID}
FriendsList	-	{AppUser_ID1} + {AppUser_ID2}	{AppUser_ID1}, {AppUser_ID2}
Wishlist	-	{AppUser_ID} + {Software_ID}	{AppUser_ID}, {Software_ID}
Purchase_Returns	-	{Purchases_ID} + {AppUser_ID}	{Purchases_ID}, {AppUser_ID}

Diagrama Relacional



Base de Dados - SQL

SQL DDL --> Ficheiro DDL.sql

Criação de 19 tabelas necessárias para a base de dados. Chaves primárias e CHECKs definidos na criação da tabela, chaves estrangeiras e sua integridade definidas em constraints após criação das tabelas.

SQL Dataset --> Ficheiro DataSet.sql

Todas as tabelas são populadas com dados necessários para o teste e uso da base de dados e interface. Os dados introduzidos cumprem todas as restrições de integridade e seguem a lógica inerente ao funcionamento da base de dados do tema escolhido.

SQL Views --> Ficheiro Views.sql

No total foram criadas 8 views para a adaptação e apresentação de dados necessários para a gestão de aspetos mais gerais da aplicação.

A seguir é apresentada uma breve explicação do funcionamento de cada uma delas:

- **proj.Show_All_Store_Games** – mostra todos os jogos que se encontram disponíveis na loja.
- **Proj.Show_All_Store_Tools** – mostra todas as aplicações que se encontram disponíveis na loja.
- **proj.Show_Tools_Can_Add_Store** – mostra todos os jogos que existem, não estão na loja, mas que podem ser adicionados a esta.
- **proj.Show_Games_Can_Add_Store** – mostra todas as aplicações que existem, não estão na loja, mas que podem ser adicionadas a esta.
- **proj.Show_All_Allowed_Publishers** – mostra todas as publicadoras autorizadas na loja.
- **proj.Show_All_NotAllowed_Publishers** – mostra todas as publicadoras não autorizadas na loja.
- **proj.Show_All_Users** – mostra todos os utilizadores existentes.
- **proj.Show_All_Items** – mostra todos os itens existentes.

SQL Stored Procedures --> Ficheiro Stored_Procedures.sql

No total foram criadas 37 SP para realizar ações mais complexas, contendo Transactions onde é necessário garantir o sucesso de mais que uma operação de UPDATE/DELETE/INSERT. A seguir é apresentada uma breve explicação do funcionamento de cada uma delas:

- **proj.searchAppUser(@StringFind) –** usada para realizar queries na tabela proj.AppUser. Devolve um record-set com todos os utilizadores onde Email, Fname ou Lname contém @StringFind
- **proj.searchNonFriend(@AppUser_ID, @StringFind)** – usada para realizar queries na tabela obtida pela UDF proj.getNotFriendsWithByUserID (tabela com apenas utilizadores não amigos de @AppUser_ID). Devolve um record-set com todos os utilizadores não amigos onde Email, Fname ou Lname contém @StringFind.
- **proj.searchPublisher(@StringFind)** – usada para realizar queries na tabela proj.Publisher. Devolve um record-set com todas as publicadoras onde NIPC ou Legal_Name contém @StringFind.
- **proj.searchGame(@Publisher_NIPC, @StringFind)** – usada para realizar queries na tabela obtida pela UDF proj.GetGamesByPublisher (tabela com apenas os jogos publicados por @Publisher_NIPC). Devolve um record-set com todos os jogos desta publicadora onde ID ou Official_Name contém @StringFind.
- **proj.searchTool(@Publisher_NIPC, @StringFind)** - usada para realizar queries na tabela obtida pela UDF proj.GetToolsByPublisher (tabela com apenas os jogos publicados por @Publisher_NIPC). Devolve um record-set com todas as aplicações onde ID ou Official_Name contém @StringFind.
- **proj.searchItem(@StringFind)** – usada para realizar queries na tabela proj.Item. Devolve um record-set com todos os itens onde UUID ou Item_Name contém @StringFind.
- **proj.searchRandomUserID(@User_ID) –** usada para obter o ID de um utilizador aleatório.
- **proj.createAppUser(@Email, @Fname, @Lname, @Birthdate, @Sex, @Street, @Postcode, @City, @Country)** - usada para criar um novo Utilizador.

- **proj.createPublisher(@NIPC, @Legal_Name, @Street, @Postcode, @City, @Country, @Found_Date, @IsAllowed)** - usada para criar uma nova Publicadora.
- **proj.createGame(@Official_Name, @Release_Date, @Publisher_NIPC, @Age_Rating, @Brief_Description, @Game_Type, @SupportedOS)** – usada para criar um novo Jogo. Insere primeiro o registo na tabela proj.Software, após isso, é recuperado o ID desta nova inserção, e é inserido na tabela proj.Game o registo usando esse ID.
  Finalmente insere na tabela proj.SoftOS e proj.Game_Type os tuplos respetivos, usando para isso o seguinte código:

```
DECLARE @Separador AS NVARCHAR(1);
SET @Separador = ',';

DECLARE @SP1 INT;
DECLARE @TypeValue NVARCHAR(255);

WHILE PATINDEX('%' + @Separador + '%', @Game_Type ) <> 0
    BEGIN
        SELECT  @SP1 = PATINDEX('%' + @Separador + '%', @Game_Type);
        SELECT  @TypeValue = LEFT(@Game_Type , @SP1 - 1);
        SELECT  @Game_Type = STUFF(@Game_Type, 1, @SP1, '');
        INSERT INTO proj.Game_Type VALUES (@Soft_ID, @TypeValue);
    END
```

A Stored Procedure recebe @Game_Type e @SupportedOS, que são do tipo NVARCHAR. Usando como exemplo @Game_Type, assumindo que é recebido como <Shooter, RPG, Story, Survival>:

1. **@Separador é declarado como sendo uma virgula**
2. **@TypeValue é declarado como sendo um NVARCHAR**
3. **Usando a função PATINDEX (que retorna a posição inicial, começando em 1, da primeira ocorrência de @Separador em @Game_Type), é criado um ciclo WHILE que corre enquanto for encontrada uma virgula em @Game_Type.**
4. **A @SP1 é dado o valor da posição da primeira ocorrência de uma virgula em @Game_Type.**
5. **Usando a função LEFT, extraímos os primeiros @SP1-1 caracteres da string @Game_Type, ou seja, no exemplo dado, é extraído o valor <Shooter>**
6. **Usando a função STUFF, substituímos em @Game_Type os caracteres da posição 1 até a posição @SP1 por uma string vazia, ou seja, no exemplo, é substituído <Shooter,> por ´´, ficando @Game_Type com o valor <RPG, Story, Survival>**
7. **É inserido na tabela proj.Game_Type o tuplo (@Soft_ID, @Type_Value), ou seja, no exemplo dado, é inserido (@Soft_ID, Shooter)**
8. **Ciclo é repetido, até não existir mais valores em @Game_Type**

Sendo @Game_Type e @SupportedOS atributos multi-valor, é assim possível receber vários valores ao mesmo tempo e separá-los, inserindo na tabela correspondente os vários tuplos.

- **proj.createTool(@Official_Name, @Release_Date, @Publisher_NIPC, @Current_Version, @Brief_Description, @SupportedOS)** - usada para criar uma nova Aplicação. Insere primeiro o registo na tabela proj.Software, após isto, é recuperado o ID desta nova inserção, e é inserido na tabela proj.Tool o registo usando esse ID.
  Finalmente insere na tabela proj.SoftOS os tuplos respetivos, usando para isso o código explicado na página anterior.
- **proj.createItem(@Item_Name, @Rarity, @Market_Value, @Category, @Game_ID, @AppUser_ID)** – usada para criar um novo Item. Se @Category do item especificar que este pode ser vendido (categoria 2, 3, 4), é criado um novo item com Market_Value = @Market_Value, senão (se for categoria 1), é criado com Market_Value = NULL. Finalmente é inserido no inventário de um utilizador aleatório, sendo que @AppUser_ID é passado à função no código C# e resulta do uso da SP proj.searchRandomUser.
- **proj.deletePublisher(@Publisher_NIPC)** – usada para apagar uma Publicadora. É apagado primeiro o software da publicadora de todas as listas de desejos, e só depois é apagada a publicadora. Os restantes delete são garantidos pelas contraints criadas nas tabelas, pelo que não é necessário mais nenhum DELETE explicito nesta SP.
- **proj.deleteAppUser(@AppUser_ID)** – usada para apagar um Utilizador. É apagado primeiro o utilizador e só depois os seus itens. Os restantes deletes são garantidos pelas contraints criadas nas tabelas, pelo que não é necessário mais nenhum DELETE explicito nesta SP.
- **proj.deleteGame(@Software_ID)** – usada para apagar um Jogo. Os restantes deletes são garantidos pelas contraints criadas nas tabelas e pelo Trigger proj.deleteGameSoft, pelo que não é necessário mais nenhum DELETE explicito nesta SP.
- **proj.deleteTool(@Software_ID)** – usada para apagar uma aplicação. Os restantes delete são garantidos pelo Trigger proj.deleteToolSoft, pelo que não é necessário mais nenhum DELETE explicito nesta SP.
- **proj.deleteItem(@UUID, @AppUser_ID)** – usada para apagar um Item. Primeiro é recuperado o Market_Value do item com @UUID. Se este for NULL, apenas é apagado o item. Se não, é apagado o item e o atributo Balance do utilizador que possui este item é atualizado, sendo-lhe somado o

Market_Value. Os restantes deletes são garantidos pelas contraints criadas nas tabelas, pelo que não é necessário mais nenhum DELETE explicito nesta SP.

- **proj.addStore_Software(@Software_ID, @Price)** – usada para adicionar Software existente à loja.
- **proj.addAuthorizationList(@Publisher_NIPC)** – usada para autorizar uma Publicadora a vender na Loja. Primeiro é atualizado o atributo IsAllowed da publicadora para `'1'` (indicando que passou a estar autorizada), e só depois é adicionada à tabela proj.Authorization_List.
- **proj.addBalance(@AppUser_ID, @Value)** – usada para adicionar Saldo a um utilizador. O atributo Balance é atualizado, tendo como novo valor = Balance antigo+@Value.
- **proj.addFriend(@AppUser_ID1, @AppUser_ID2)** – usada para adicionar um utilizador à lista de amigos.
- **proj.addWishlist(@AppUser_ID, @Software_ID)** – usada para adicionar um Software à lista de desejos de um utilizador.
- **proj.removeStoreSoftware(@Software_ID)** – usada para remover Software da loja. É primeiro removido o software da lista de desejos de todos os utilizadores, e só depois é removido da loja.
- **proj.removeAuthorizationList(@Publisher_NIPC)** – usada para remover uma Publicadora da lista de autorização. É primeiro removido o software da publicadora da lista de desejos de todos os utilizadores, seguido da sua remoção da loja. Depois é removida a publicadora da lista de autorização e finalmente é atualizado o atributo IsAllowed para `'0'`(indicando que passou a não estar autorizada).
- **proj.removeFriend(@AppUser_ID1, @AppUser_ID2)** – usada para remover um utilizador da lista de amigos de outro utilizador.
- **proj.removeWishlist(@AppUser_ID, @Item_UUID)** – usada para remover software da lista de desejos de um utilizador.
- **proj.removeItemFromMarket(@Item_UUID)** – usada para remover Itens do mercado da comunidade. É primeiro removido o item do mercado e depois é atualizado o atributo ForSale para `'N'`, (indicando que passou a não está venda).

- **proj.editSoftwarePrice(@Software_ID, @New_Price)** – usada para editar o preço de um Software na loja.
- **proj.editAppUser(@AppUser_ID, @Email, @Fname, @Lname, @Birthdate, @Sex, @Street, @Postcode, @City, @Country)** – usada para editar um Utilizador existente.
- **proj.editPublisher(@CurrentNIPC, @NewNIPC, @Legal_Name, @Street, @Postcode, @City, @Country, @IsAllowed)** – usada para editar uma Publicadora existente.
- **proj.editGame(@Software_ID, @Official_Name, @Release_Date, @Age_Rating, @Brief_Description, @Game_Type, @SupportedOS)** – usada para editar um Jogo existente.
- **proj.editTool(@Software_ID, @Official_Name, @Release_Date, @Current_Version, @Brief_Description, @SupportedOS)** – usada para editar uma Aplicação existente.
- **proj.editItem(@UUID, @Item_Name, @Rarity, @Market_Value, @Category, @Game_ID, @AppUser_ID)** – usada para editar um Item existente. Se a categoria for alterada para '1' (não pode ser vendido), Market_Value do item passa a ser NULL, independentemente do valor passado em @Market_Value. Se não, o item é editado normalmente.
- **proj.buySoftware(@AppUser_ID, @Software_ID)** – usada para realizar uma compra na Loja. Se o Utilizador não tiver saldo suficiente para comprar o Software, é levantado um erro indicando esse problema. Caso contrário a compra é concretizada, ficando o registo na tabela proj.Purchases. O atributo Balance do utilizador é atualizado, sendo removido o montante correspondente ao Market_Value do software comprado, e finalmente é removido o software da lista de desejos (caso este la esteja).
- **proj.returnSoftware(@AppUser_ID, @Software_SKU) –** usada para devolver Software comprado. Software apenas pode ser devolvido se o tempo de compra não ultrapassar 3 dias, sendo que a garantia desta restrição é feita ao nível da aplicação e não da base de dados. É, portanto, realizada a devolução, ficando o registo na tabela proj.Purchase_Returns. O atributo Balance do utilizador é atualizado, sendo-lhe somado o valor gasto no software no momento da compra (este valor está guardado na tabela proj.Purchases).

- **proj.buyMarketItem(@Buyer_ID, @Item_UUID)** – usada para comprar Items no Mercado da Comunidade. Se o Utilizador não tiver saldo suficiente para comprar o Item, é levantado um erro indicando esse problema. Caso contrário é removido o item do inventário do vendedor e colocado no inventário do comprador. Seguidamente são atualizados os atributos Balance de ambos e o registo do item é apagado do mercado. Finalmente é atualizado o atributo ForSale do item para `'N'` (indicando que passou a não estar à venda).
- **proj.sellMarketItem(@Seller_ID, @Item_UUID)** – usada para vender Items no Mercado da Comunidade. Se a categoria do item não permitir a venda, é levantado um erro indicando esse problema. Caso contrário, é inserido o item no mercado, ficando esse registo na tabela proj.Market_Items, e é atualizado o atributo ForSale para `'Y'` (indicando que passou a estar à venda).

SQL User Defined Functions --> Ficheiro UDFs.sql

No total foram criadas 24 UDFs para realizar consultas mais complexas (uso frequente de JOINs) e obter dados mais completos. A seguir é apresentada uma breve explicação do funcionamento de cada uma delas:

- **proj.getOwnedSoftwareByUserID(@AppUser_ID)** – devolve uma tabela contendo informação sobre software que foi comprado e não devolvido por um utilizador.
- **proj.getPurchasesByUserID(@AppUser_ID)** – devolve uma tabela com todas as compras feitas por um utilizador.
- **proj.getPurchaseReturnsByUserID(@AppUser_ID)** – devolve uma tabela com todas as devoluções feitas por um utilizador.
- **proj.getFriendsListByUserID(@AppUser_ID)** – devolve uma tabela com informação sobre todos os amigos de um utilizador.
- **proj.getNotFriendsWithByUserID(@AppUser_ID)** – devolve uma tabela com informação sobre todos os utilizadores que não são amigos de um escolhido utilizador(@AppUser_ID)
- **proj.getWishlistByUserID(@AppUser_ID)** – devolve uma tabela com informação sobre o software que está na lista de desejos de um utilizador.
- **proj.getInventoryByUserID(@AppUser_ID)** – devolve uma tabela com informação sobre os itens que um utilizador tem.
- **proj.getMarketByUserID(@AppUser_ID)** – devolve uma tabela com os itens no mercado disponíveis para compra a um utilizador.
- **proj.getItemsForSaleByUserID(@AppUser_ID)** – devolve uma tabela com os itens que um utilizador tem a venda no mercado.
- **proj.getGamesByPublisher(@Publisher_NIPC)** – devolve uma tabela com informação sobre todos os jogos de uma publicadora.
- **proj.getToolsByPublisher(@Publisher_NIPC)** – devolve uma tabela com informação sobre todas as aplicações de uma publicadora.
- **proj.getGameSales()** – devolve uma tabela com o número de vendas por jogo.
- **proj.getToolSales()** – devolve uma tabela com o número de vendas por aplicação.

- **proj.getPublisherSales()** – devolve uma tabela com o número de vendas por publicadora.
- **proj.getMostSupportedOS()** – devolve uma tabela com o/os Sistemas Operativos mais suportados pelo software da loja.
- **proj.getAverageGamePrice()** – devolve um decimal com o preço médio dos jogos da loja.
- **proj.getAverageToolPrice()** – devolve um decimal com o preço médio das aplicações da loja.
- **proj.getSexRepresentation()** – devolve uma tabela com a representação de sexo dos utilizadores em %.
- **proj.getUsersByCountry()** – devolve uma tabela com o número de utilizadores por país.
- **proj.getUserStoreStatistics()** – devolve uma tabela com informação sobre os valores gastos em loja por cada utilizador.
- **proj.getWishedSoftware()** – devolve uma tabela com o número de utilizadores que desejam cada software.
- **proj.getTotalMoneySpent()** – devolve um decimal com o montante total gasto em loja por todos os utilizadores.
- **proj.getAverageNumberFriends()** – devolve um inteiro com o número médio de amigos dos utilizadores.
- **proj.getAverageUserAge()** – devolve um inteiro com a idade média dos utilizadores.

SQL Triggers --> Ficheiro Triggers.sql

No total foram criados 2 Triggers DML do tipo **AFTER DELETE**. Visto que as tabelas proj.Game e proj.Tool são especializações da tabela proj.Software, a criação destes dois triggers foi considerada a melhor opção para apagar registos da tabela proj.Software.
A seguir é apresentada uma breve explicação do funcionamento de cada um deles:

- **TRIGGER deleteGameSoft ON proj.Game AFTER DELETE** – É obtido o @Software_ID do registo apagado. Depois é verificada a existência desse registo nessa tabela e, caso o delete tenha sido bem sucedido e o registo já não exista, é apagado o registo correspondente na tabela proj.Software usando @Software_ID.
- **TRIGGER deleteToolSoft ON proj.Tool AFTER DELETE** – É obtido o @Software_ID do registo apagado. Depois é verificada a existência desse registo nessa tabela e, caso o delete tenha sido bem sucedido e o registo já não exista, é apagado o registo correspondente na tabela proj.Software usando @Software_ID.

SQL Indexes --> Ficheiro Indexes.sql

No total foram criados 6 indexes em atributos frequentemente usados em consultas. Os indexes **UNIQUE** foram criados em atributos que também são **UNIQUE** nas tabelas.
A seguir é apresentada uma breve explicação de cada um deles:

- **UNIQUE NONCLUSTERED INDEX IXSKU ON proj.Purchases(SKU)** – Criado pois atributo SKU é bastante usado na devolução de software.
- **UNIQUE NONCLUSTERED INDEX IXPubName ON proj.Publisher(Legal_Name)** – Criado pois é frequente a pesquisa de Publicadoras por nome.
- **UNIQUE NONCLUSTERED INDEX IXSoftName ON proj.Software(Official_Name)** – Criado pois é frequente a pesquisa de Software por nome.
- **NONCLUSTERED INDEX IXIsAllowed ON proj.Publisher(IsAllowed)** – Criado pois o atributo IsAllowed é bastante usado na construção das views.
- **NONCLUSTERED INDEX IXUserQuery ON proj.Appuser(Email, Fname, Lname)** – Criado pois é frequente a pesquisa de utilizadores por estes atributos.
- **NONCLUSTERED INDEX IXItemName ON proj.Item(Item_Name)** – Criado pois é frequente a pesquisa de itens por nome.

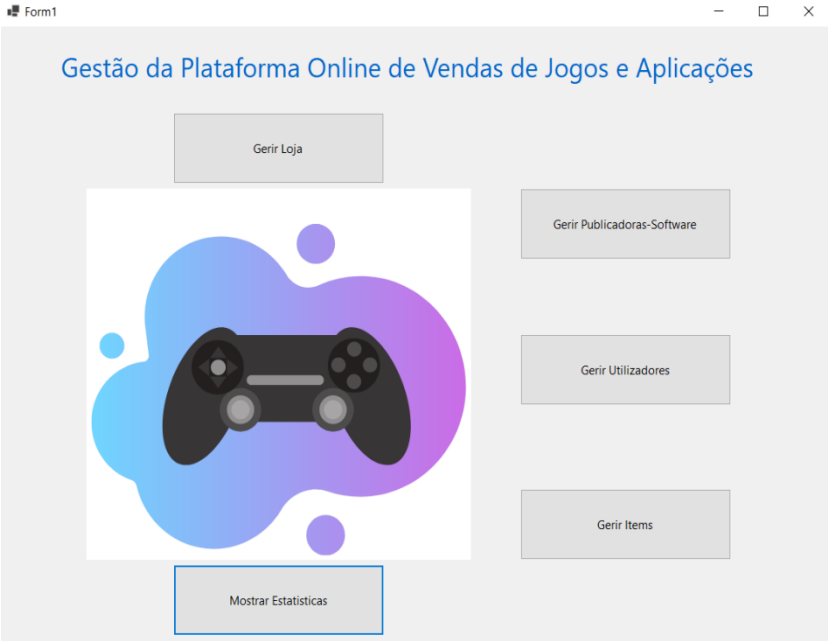
Interface – C#

A interface foi implementada em C# usando Windows Forms. No total foram criados 9 formulários, a maioria deles com várias tabs.

DB Connection String --> linha 15 do ficheiro Form1.cs

A seguir é apresentada uma breve explicação do funcionamento de cada formulário:

Form1



Form da página inicial, contém os botões que permitem navegar no resto da aplicação.

Botão ***Mostrar Estatísticas*** -> Abre Form2
Botão ***Gerir Loja*** -> Abre Form3
Botão ***Gerir Utilizadores*** -> Abre Form4
Botão ***Gerir Publicadoras-Software*** -> Abre Form5
Botão ***Gerir Itens*** -> Abre Form9

Form2

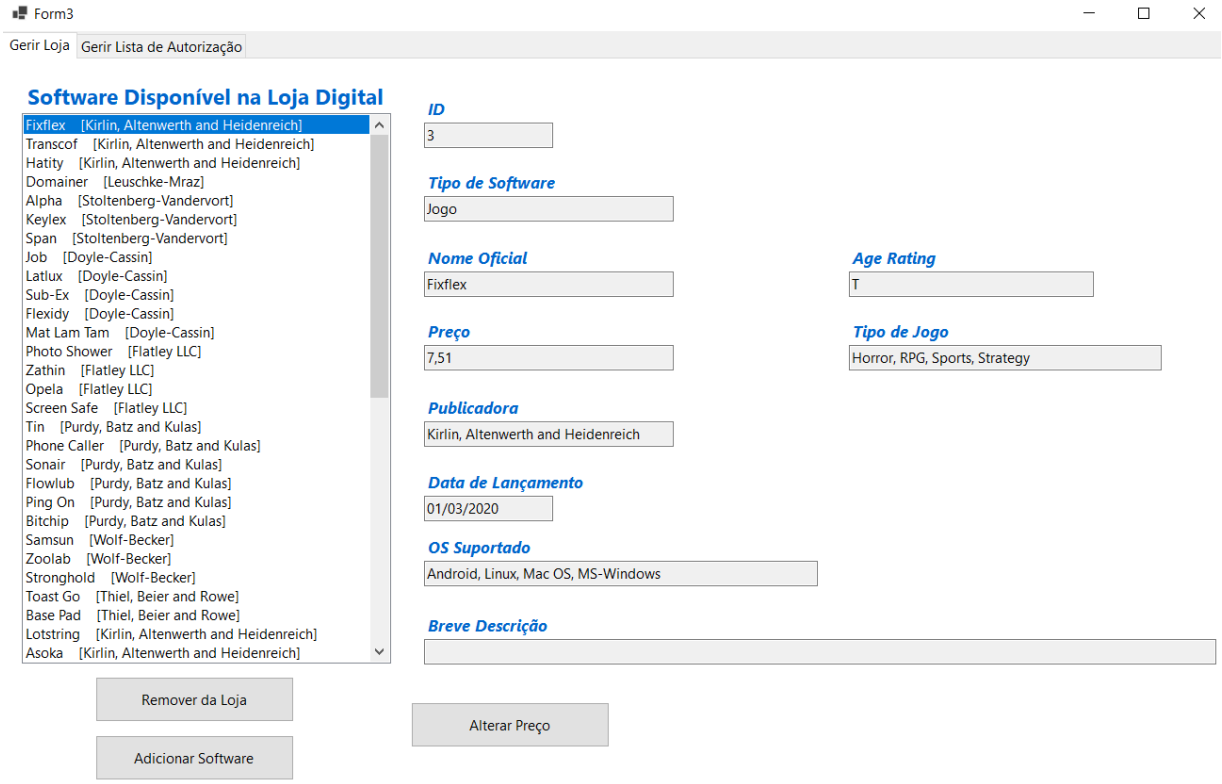
Form para ver estatísticas acerca da base de dados.

Está dividido em 2 tabs:

- **Estatísticas da Loja** – contém estatísticas relacionadas com a parte da loja.

- **Estatísticas dos Utilizadores** – contém estatísticas relacionadas com a parte dos utilizadores.

Form3

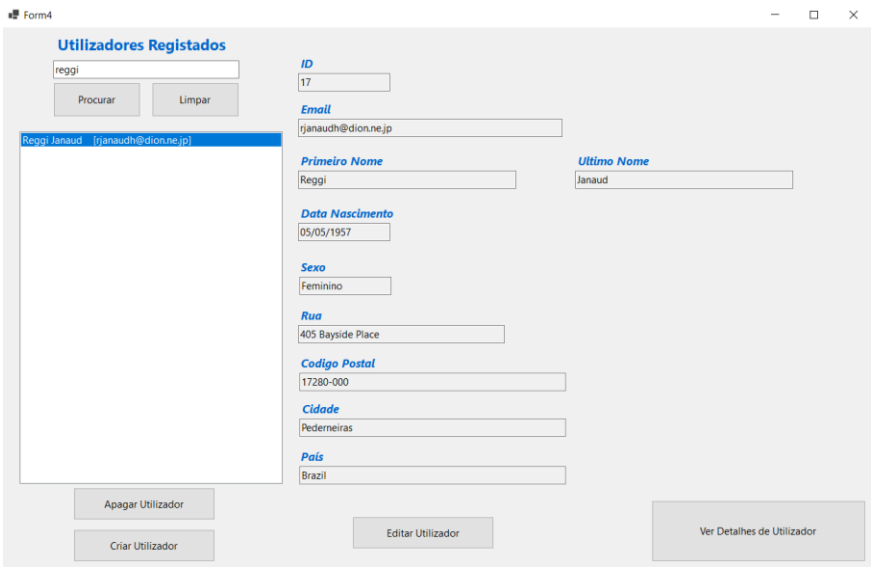


<u>Form para gerir aspetos relacionados com a loja.</u>

Está dividido em 2 tabs:

- ***Gerir Loja*** – permite editar o preço de produtos, assim como adicionar/remover software da loja.
- ***Gerir Lista de Autorização*** – permite ver as publicadoras que estão autorizadas/não autorizadas na loja, assim como adicionar/remover tais publicadoras da lista de autorização.

Form 4

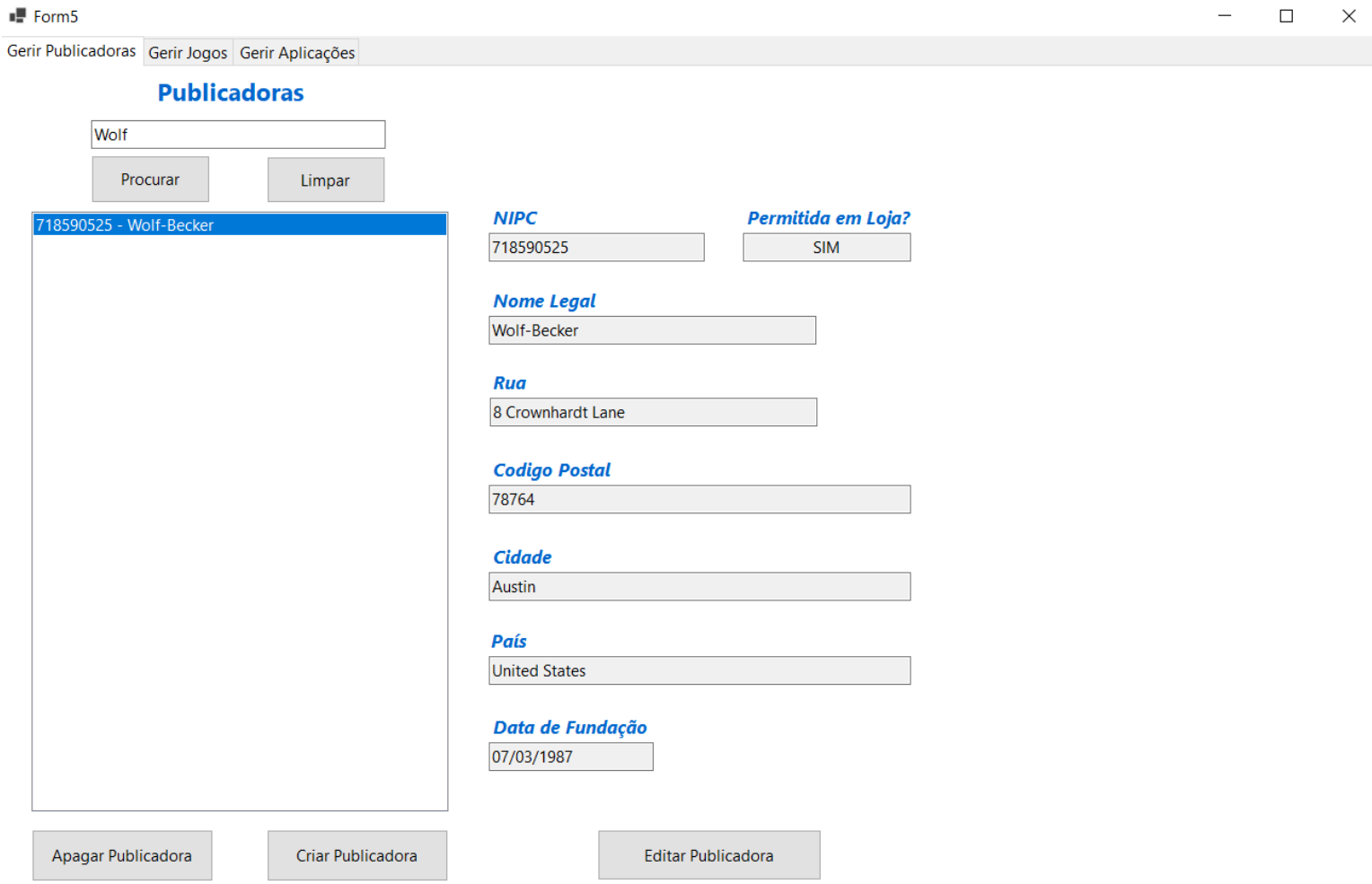


<u>Form para gestão geral de utilizadores da plataforma.</u>

Permite criar/apagar/editar utilizadores, assim como pesquisar utilizadores existentes por Email, Fname ou Lname.

Botão ***Ver Detalhes do Utilizador*** -> Abre Form6

Form5

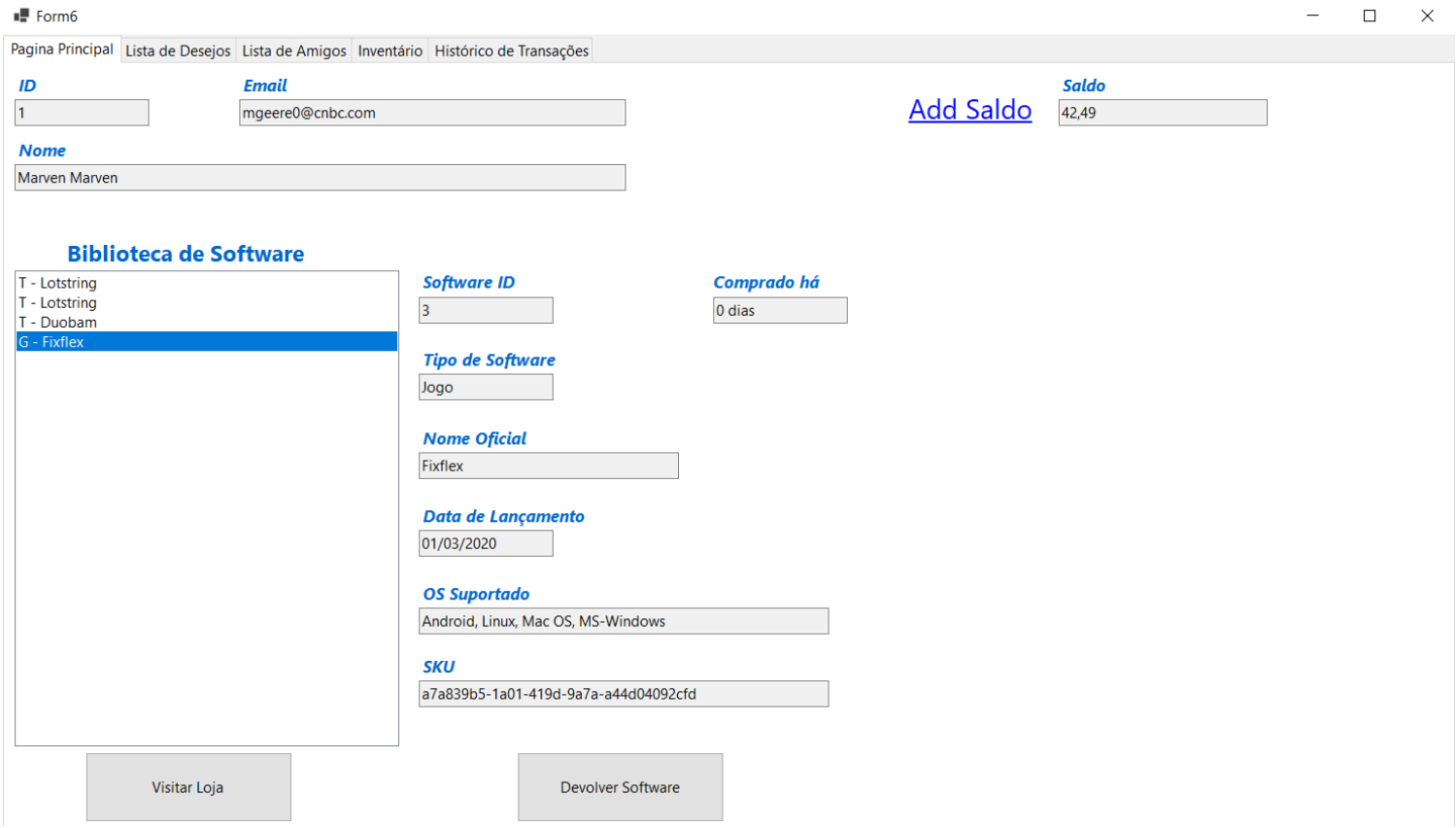


Form para gerir publicadoras e seu software.

Está divido em 3 tabs:

- ***Gerir Publicadoras*** – permite criar/apagar/editar publicadoras, assim como pesquisar publicadoras existentes por NIPC ou Legal_Name.
- ***Gerir Jogos*** - permite criar/apagar/editar jogos, assim como pesquisar jogos existentes de uma escolhida publicadora por ID ou Official_Name.
- ***Gerir Aplicações*** - permite criar/apagar/editar aplicações, assim como pesquisar aplicações existentes de uma escolhida publicadora por ID ou Official_Name.

Form6



Form para gerir detalhes da conta de um utilizador individual.

Está divido em 5 tabs:

- ***Página Principal*** – mostra a biblioteca do utilizador. Permite visitar a loja e comprar software ou então adicioná-lo à lista de desejos. Também permite devolver software que tenha sido comprado há menos de 3 dias.

  Link ***Add Saldo*** – Abre Form7.

- ***Lista de Desejos*** – mostra a lista de desejos do utilizador. Permite remover software desta ou então comprá-lo diretamente.

  Link ***Add Saldo*** – Abre Form7.

- ***Lista de Amigos*** – mostra a lista de amigos do utilizador, assim como os utilizadores não amigos. Permite adicionar/remover utilizadores da lista de amigos, assim como pesquisar utilizadores não amigos por Email, Fname ou Lname.

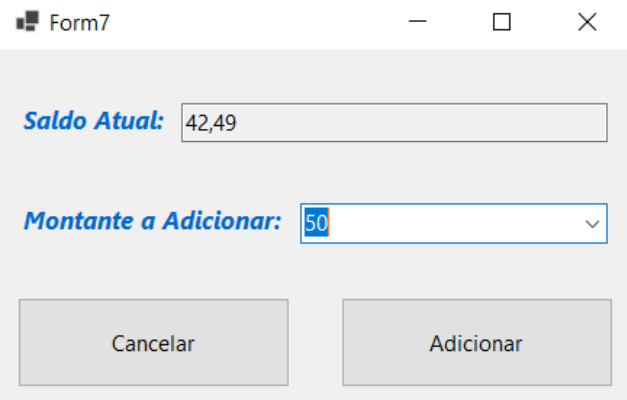
- ***Inventário*** – mostra o inventário de itens do utilizador. Permite vender/comprar itens no mercado.

  Link ***Add Saldo*** – Abre Form7.

  Botão **Gerir Itens Colocados À Venda** – Abre Form8.

- ***Histórico de Transações*** – mostra as compras e as devoluções do utilizador na loja.

Form7

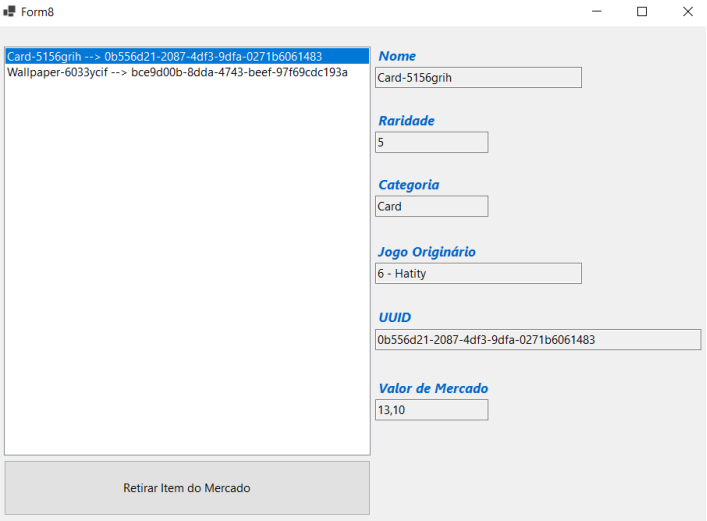


Form para gerir saldo do utilizador.

Permite carregar saldo com valores de uma lista pré definida:

- 5, 10, 25, 50, 100, 250, 500

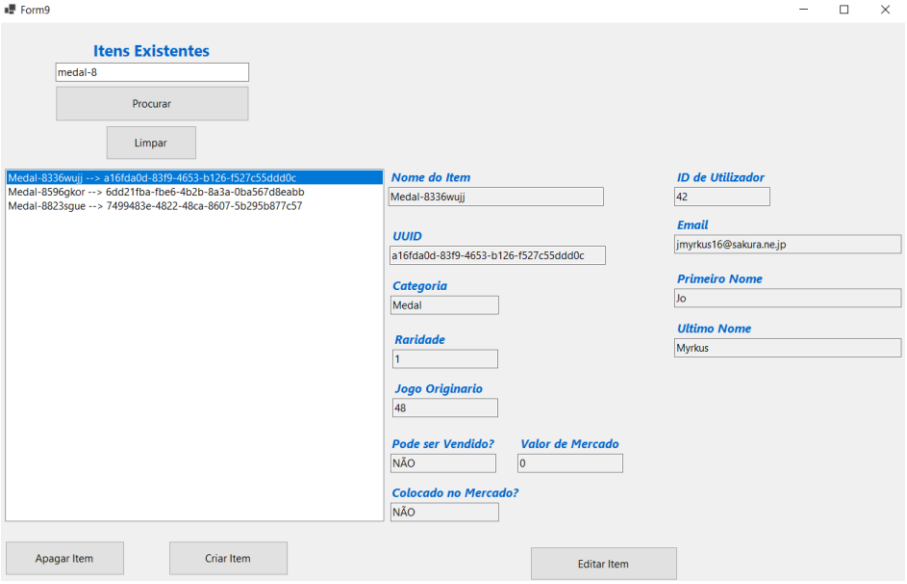
Form8



Form para gerir itens que um utilizador tem à venda no mercado.

Permite ver quais os itens que tem a venda no mercado, assim como os remover deste.

Form9



Form para gerir itens da plataforma.

Permite criar/apagar/editar itens, assim como procurar itens existentes por UUID ou Item_Name.