**Toegepaste Informatica**

**Documentatie casus: Vintage Items**

*OO-AD en Database Development*

**Jan Dammans**

(r0796935)

Eerste bachelor – Toegepaste Informatica

Klasgroep : 5

**OO-AD en Database Development**

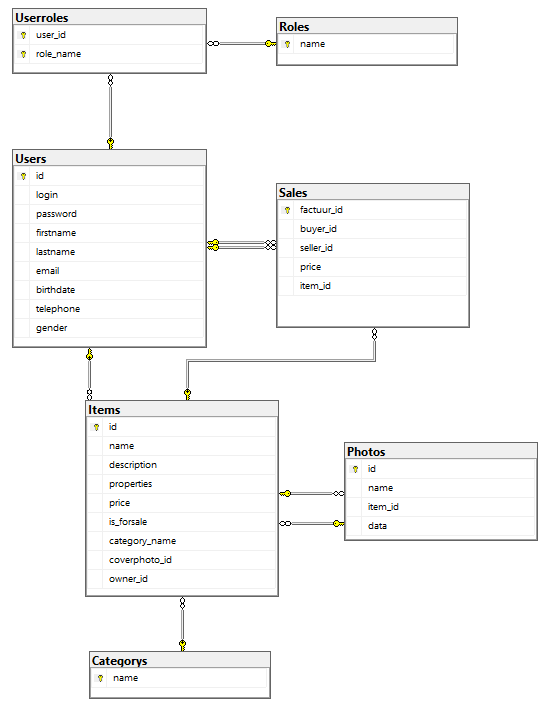
Rogier van de Linde

Academiejaar 2019-2020

# Database

## Database diagram

#### Schema



#### Constraints

##### Userroles

FK: role\_name PK:Roles.name

FK: user\_id PK: Users.id

##### Items

FK: owner\_id Pk: Users.id

FK: coverphoto\_id PK: Photos.id

FK: category\_name PK:Categories.name

##### Photos

FK: item\_id PK: Items.id

##### Sales

FK: buyer\_id PK: Users.id

FK: seller\_id PK: Users.id

FK: item\_id PK: Items.id

## Views

### vwForSaleItems

CREATE OR ALTER view [dbo].[vwForSaleItems]

as

select id,name,description,properties,dbo.fnAddBTW(price) as price,is\_forsale,category\_name,coverphoto\_id,owner\_id

from Items

where is\_forsale='Y'

### GO

### vwHistory

CREATE OR ALTER view [dbo].[vwHistory]

as

select factuur\_id,B.firstname as Bfirst,B.lastname as Blast,Sl.firstname as Sfirst,Sl.lastname as Slast,price,item\_id

from Sales S inner join Users B on B.id=S.buyer\_id

inner join Users Sl on S.seller\_id=Sl.id

GO

### vwLoginPassword

CREATE OR ALTER view [dbo].[vwLoginPassword]

as

select id,login,password, role\_name,firstname,gender, lastname, birthdate,email,telephone

from Users U inner join Userroles Us on U.id=Us.user\_id

GO

### vwUsers

CREATE OR ALTER view [dbo].[vwUsers]

as

select \*

from Users U left join Userroles Ur on U.id=Ur.user\_id

GO

## Stored Procedures

### spAddItem

ALTER PROCEDURE [dbo].[spAddItem](@name nvarchar(200), @description nvarchar(2000), @properties nvarchar(2000), @price decimal,

@isforsale nvarchar(1), @category\_name nvarchar(50), @owner\_id int, @namePhoto nvarchar(2000), @data image)

AS

BEGIN

/\*insert item\*/

INSERT INTO Items(name,description,properties,price,is\_forsale,category\_name,owner\_id)

VALUES (@name, @description, @properties, @price, @isforsale, @category\_name,@owner\_id );

/\*add photo\*/

Declare @item\_id int;

set @item\_id=(select id from dbo.[Items] where [Items].[name]=@name and price=@price and [description]=@description and @isforsale=is\_forsale)

if(@namePhoto!='')

begin

INSERT INTO Photos(name,item\_id,data)

VALUES (@namePhoto, @item\_id,@data );

Declare @photoId int;

set @photoId= (select id from Photos where item\_id=@item\_id)

Update Items

SET coverphoto\_id = @photoId WHERE ID = @item\_id;

end

END

### spAddUser

ALTER PROCEDURE [dbo].[spAddUser]( @login nvarchar(50),@password nvarchar(255),@firstname nvarchar(50),@lastname nvarchar(50),@email nvarchar(255),@birthdate date,@telephone nvarchar(50),@rolename nvarchar(50),@gender nvarchar(1)) AS

BEGIN

/\*insert user data\*/

if not exists(select login from Users where login= @login)

begin

INSERT INTO [dbo].[Users](login,password,firstname,lastname,email,birthdate,telephone,gender) \_

VALUES (@login,@password,@firstname,@lastname,@email,@birthdate,@telephone,@gender);

end

Declare @id int;

set @id =(select id

from dbo.[Users]

where login=@login and password=@password)

/\*insert user role\*/

INSERT INTO [dbo].[Userroles](role\_name,user\_id) \_

VALUES (@rolename,@id);

END;

### spBuyItem

ALTER procedure [dbo].[spBuyItem] (@itemid int,@ownerid int)

as

begin

update Items

Set owner\_id = @ownerid, is\_forsale='N'

where id = @itemid

end

### spCreateLogin

ALTER procedure [dbo].[spCeateLogin]

as

Begin

DECLARE @int INT = 0;

DECLARE @sqlLogin nvarchar(2000);

DECLARE @sqlUser nvarchar(2000);

WHILE @int < (select Count(\*) from Users) or @int < (SELECT MAX(id)FROM Users)

BEGIN

DECLARE @login nvarchar(200)= (SELECT login From Users where id=@int);

DECLARE @pw nvarchar(200)= (SELECT password From Users where id=@int);

set @sqlLogin=N'CREATE LOGIN '+ @login + N' WITH PASSWORD = '''+@pw +''' ;';

set @sqlUser=N'CREATE USER '+ @login + N' FOR LOGIN ' +@login+' ;';

EXEC sp\_executesql @sqlLogin;

EXEC sp\_executesql @sqlUser;

SET @int = @int + 1;

end

END;

### spDeleteItem

ALTER procedure [dbo].[spDeleteItem](@Itemid as int)

as begin

/\*Delete in history\*/

DELETE FROM [dbo].Sales

WHERE item\_id=@Itemid

/\* drop FK \*/

ALTER TABLE Photos DROP CONSTRAINT IF EXISTS FK\_Photos\_Items;

ALTER TABLE Items DROP CONSTRAINT IF EXISTS FK\_Items\_Photos;

IF (0 <(select COUNT(item\_id) from Photos where item\_id = @Itemid))

begin

Delete from Photos where @Itemid = item\_id

end

DELETE FROM [dbo].Items

WHERE id=@Itemid

/\* re-add FK \*/

ALTER TABLE Photos ADD CONSTRAINT FK\_Photos\_Items foreign KEY(item\_id) REFERENCES Items(id);

ALTER TABLE Items ADD CONSTRAINT FK\_Items\_Photos foreign KEY(coverphoto\_id) REFERENCES Photos(id);

end

### spDeleteUser

ALTER procedure [dbo].[spDeleteUser](@id as int)

as begin

ALTER TABLE Photos DROP CONSTRAINT IF EXISTS FK\_Photos\_Items;

ALTER TABLE Items DROP CONSTRAINT IF EXISTS FK\_Items\_Photos;

ALTER TABLE Items DROP CONSTRAINT IF EXISTS FK\_Items\_Users;

/\*Delete in history\*/

Delete From Sales

where buyer\_id=@id or seller\_id=@id

/\*Delete role by user\*/

DELETE FROM [dbo].[Userroles]

WHERE user\_id=@id

/\*Delete user\*/

DELETE FROM [dbo].[Users]

WHERE id=@id

DELETE FROM [dbo].Sales

WHERE seller\_id=@id

DECLARE @itemLoop INT

DECLARE @Itemid int

DECLARE Cursor\_Items CURSOR FOR

SELECT owner\_id,id FROM Items where owner\_id=@id

OPEN Cursor\_Items

FETCH NEXT FROM Cursor\_Items INTO @itemLoop,@Itemid

WHILE @@FETCH\_STATUS = 0

BEGIN

IF (0 <(select COUNT(item\_id) from Photos where item\_id = @Itemid))

begin

Delete from Photos where @Itemid = item\_id

end

DELETE FROM [dbo].Items

WHERE id=@Itemid

FETCH NEXT FROM Cursor\_Items INTO @itemLoop,@Itemid

END

CLOSE Cursor\_Items

DEALLOCATE Cursor\_Items

/\* re-add FK \*/

ALTER TABLE Items ADD CONSTRAINT FK\_Items\_Users foreign KEY(owner\_id) REFERENCES Users(id);

ALTER TABLE Photos ADD CONSTRAINT FK\_Photos\_Items foreign KEY(item\_id) REFERENCES Items(id);

ALTER TABLE Items ADD CONSTRAINT FK\_Items\_Photos foreign KEY(coverphoto\_id) REFERENCES Photos(id);

end

### spItemsWithId

ALTER procedure [dbo].[spItemsWithId](@id int)

as

begin

select \*

from Items

where owner\_id=@id;

end

### spPhotos

ALTER procedure [dbo].[spPhotos](@id int)

as

Begin

SELECT \* FROM Photos WHERE id=@id

End

### spUpdateItem

ALTER procedure [dbo].[spUpdateItem](@id int,@name nvarchar(2000),@Description nvarchar(2000),

@properties nvarchar(2000),@price decimal(18, 0),@forSale nvarchar(1),@categoryName nvarchar(50),@coverphotoId int)

AS

BEGIN

/\*Update item\*/

UPDATE [dbo].Items \_

SET name=@name, description=@Description, properties=@properties, price=@price, is\_forsale=@forSale, category\_name=@categoryName,coverphoto\_id=@coverphotoId

where Items.id=@id;

END;

### spUpdateUser

ALTER procedure [dbo].[spUpdateUser](@id int,@login nvarchar(50),@gender nvarchar(1),@password nvarchar(255),@firstname nvarchar(50),@lastname nvarchar(50),@email nvarchar(255),@birthdate date,@telephone nvarchar(50),@rolename nvarchar(50),@oldRole\_name nvarchar(50))

AS

BEGIN

/\*Update user\*/

UPDATE [dbo].[Users] \_

SET gender=@gender, login = @login, password = @password, firstname=@firstname,lastname=@lastname,email=@email,birthdate=@birthdate,telephone=@telephone

where Users.id=@id;

/\*update userrole\*/

UPDATE [dbo].[Userroles]

Set role\_name=@rolename

where Userroles.user\_id=@id and @oldRole\_name=role\_name;

END;

### spUserWithId

ALTER procedure [dbo].[spUserWithId] (@id int)

as

select \*

from Users U inner join Userroles US on U.id=US.user\_id

where @id=U.id

## Functions

### fnAddBTW

ALTER function [dbo].[fnAddBTW](@bedrag decimal)

RETURNS DECIMAL

AS

BEGIN

/\* declare variables \*/

DECLARE @total DECIMAL;

SET @total = @bedrag\*1.21;

/\* return \*/

RETURN @total

END

## Triggers

### trItemOwnerChanged

ALTER trigger [dbo].[trItemOwnerChanged]

ON [dbo].[Items]

AFTER UPDATE

AS

INSERT INTO Sales(buyer\_id,seller\_id, price,item\_id)

SELECT inserted.owner\_id, deleted.owner\_id, inserted.price,inserted.id

FROM deleted,inserted

## Rollen

1. Db\_admin
2. Db\_buyer
3. Db\_seller

## Permissies

### db\_admin

* Items: Alter - Delete - References – Select
* Photos: Alter - Delete – Insert - References – Select – Update
* Sales: References – Select
* Userroles: Delete - Insert – Select – Update
* Users: Alter - Delete – Insert - References – Select – Update
* spAddUser: Execute
* spDeleteUser: Execute
* spUpdateUser: Execute
* vwHistory: Select
* vwLoginPassword: Select
* vwUSers: Select

### db\_buyer

* Items: Alter - Delete - References – Select
* Photos: Alter - Delete - References – Select – Update
* spDeleteItem: Execute
* spItemsWithId: Execute
* spPhotos: Execute
* spUpdateItem: Execute
* vwForSaleItems: Select

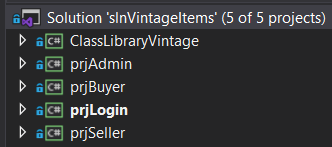
### db\_seller

* Items: Alter - Delete - Insert - References – Select - Update
* Photos: Alter - Delete - Insert - References – Select – Update
* spAddUser: Execute
* spDeleteItem: Execute
* spItemsWithId: Execute
* spPhotos: Execute
* spUpdateItem: Execute

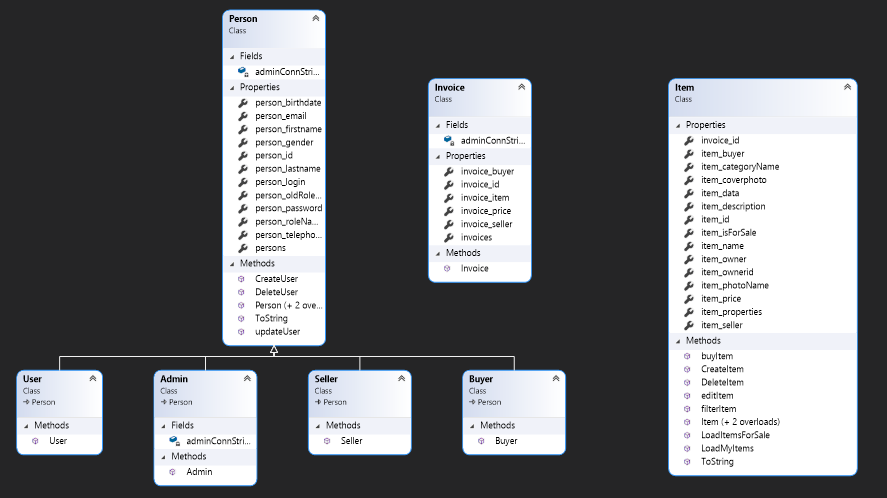
# Wpf toepassingen

## slnVintageItems

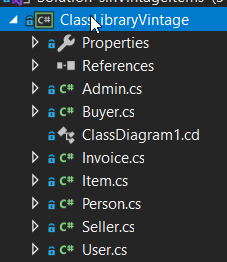
Alle wpf projecten zijn samengebundeld in de solution slnVintageITems. Ook bezit deze de class library.



### Klassediagram



### Class library



## prjLogin

Om toegang te krijgen tot de gegevens moet eerst de wpf van het project prjLogin opstarten. Afhankelijk van de rol van de user die inlogt zal het project bepalen welke wpf er zal geopend worden.

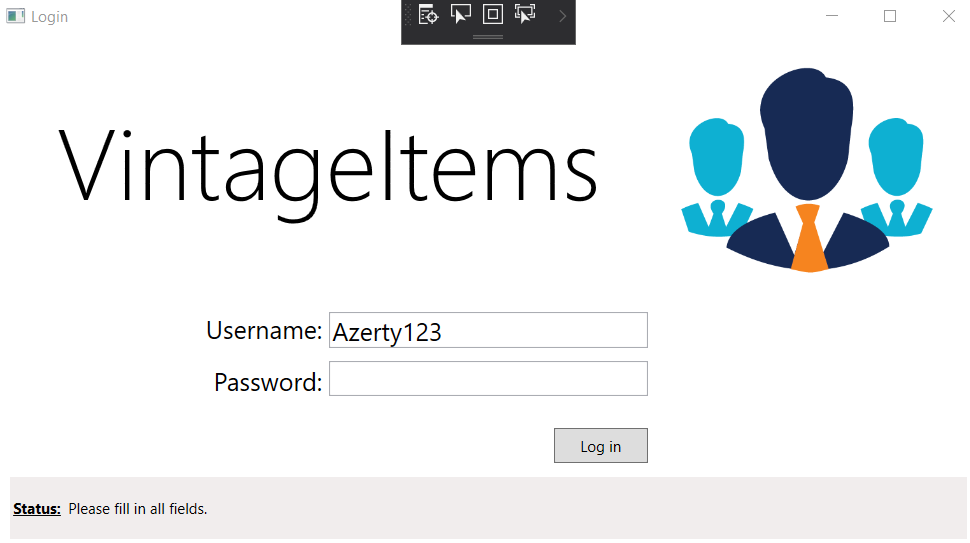
### Werking

Eerst controleert het project of alle velden ingevuld zijn. Nadien kijkt hij of het paswoord overeenkomt met de login. Indien dit niet het geval is zal het programma een error geven. Indien deze wel overeenkomen zal het programma kijken welke rol deze persoon heeft en vervolgens de juiste wpf openen.

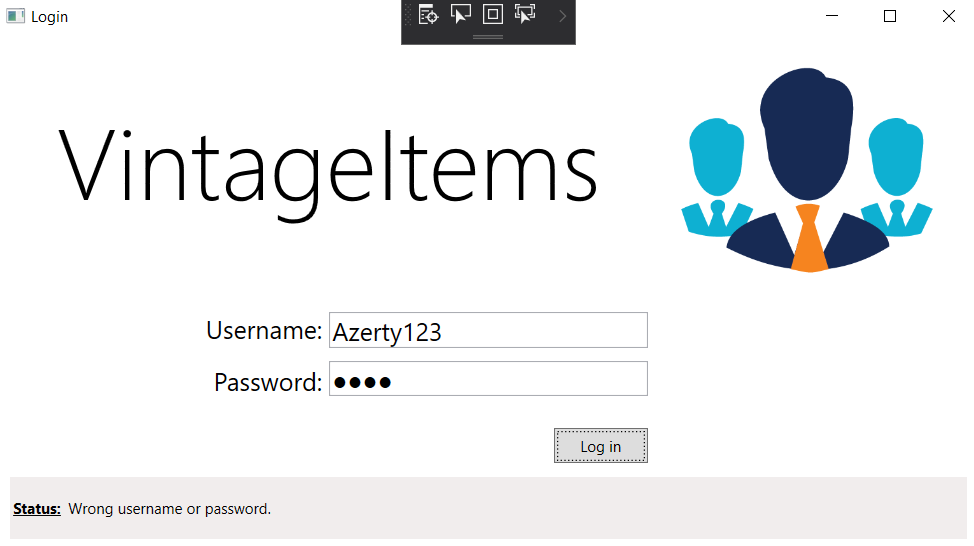
Het kan zijn dat een user meerdere toegangsmogelijkheden heeft. Indien dit het geval is zal de gebruiker gevraagd worden met welke rol hij zal willen inloggen.

### Demonstratie

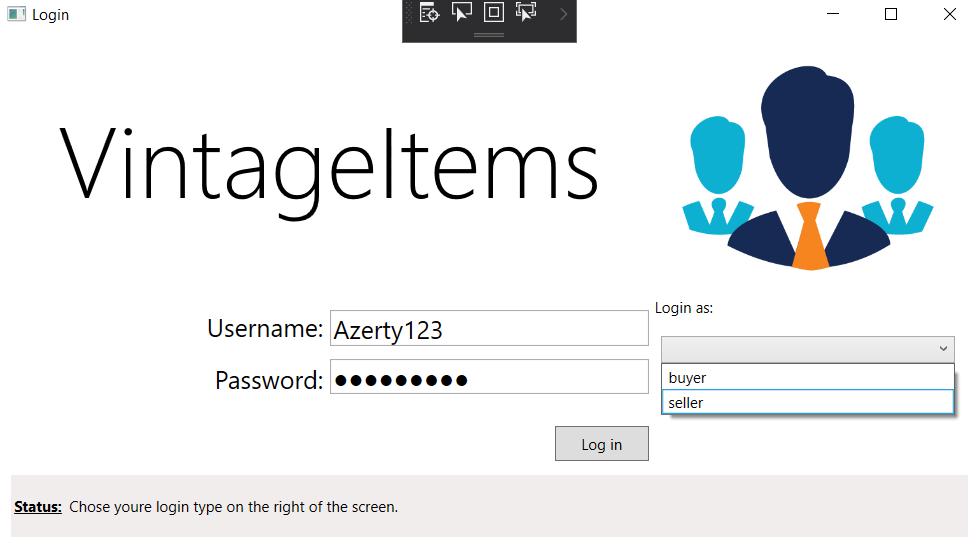
#### Niet alle velden ingevuld



#### Paswoorden komen niet overeen

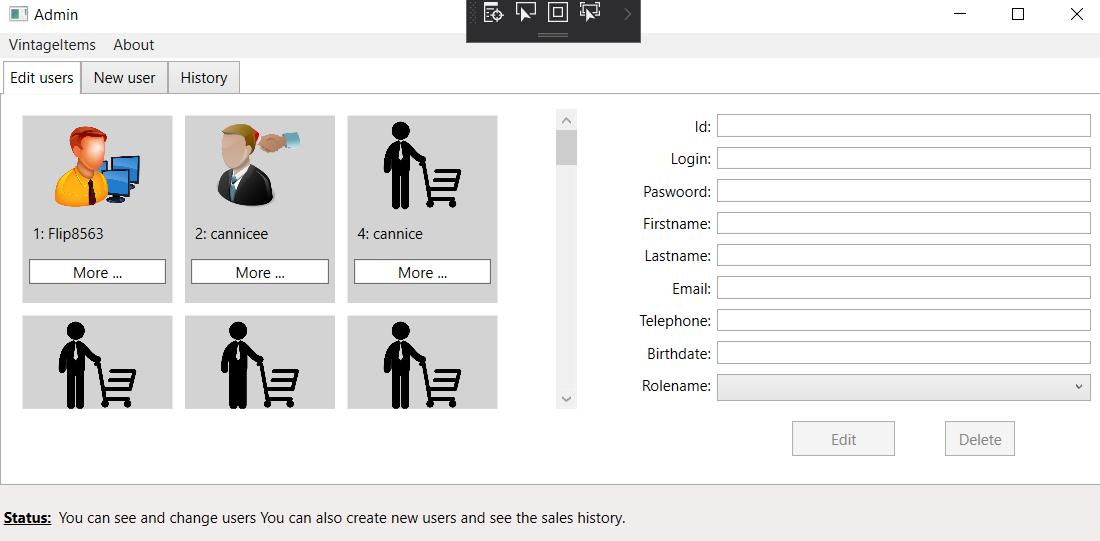


#### Meerdere rollen

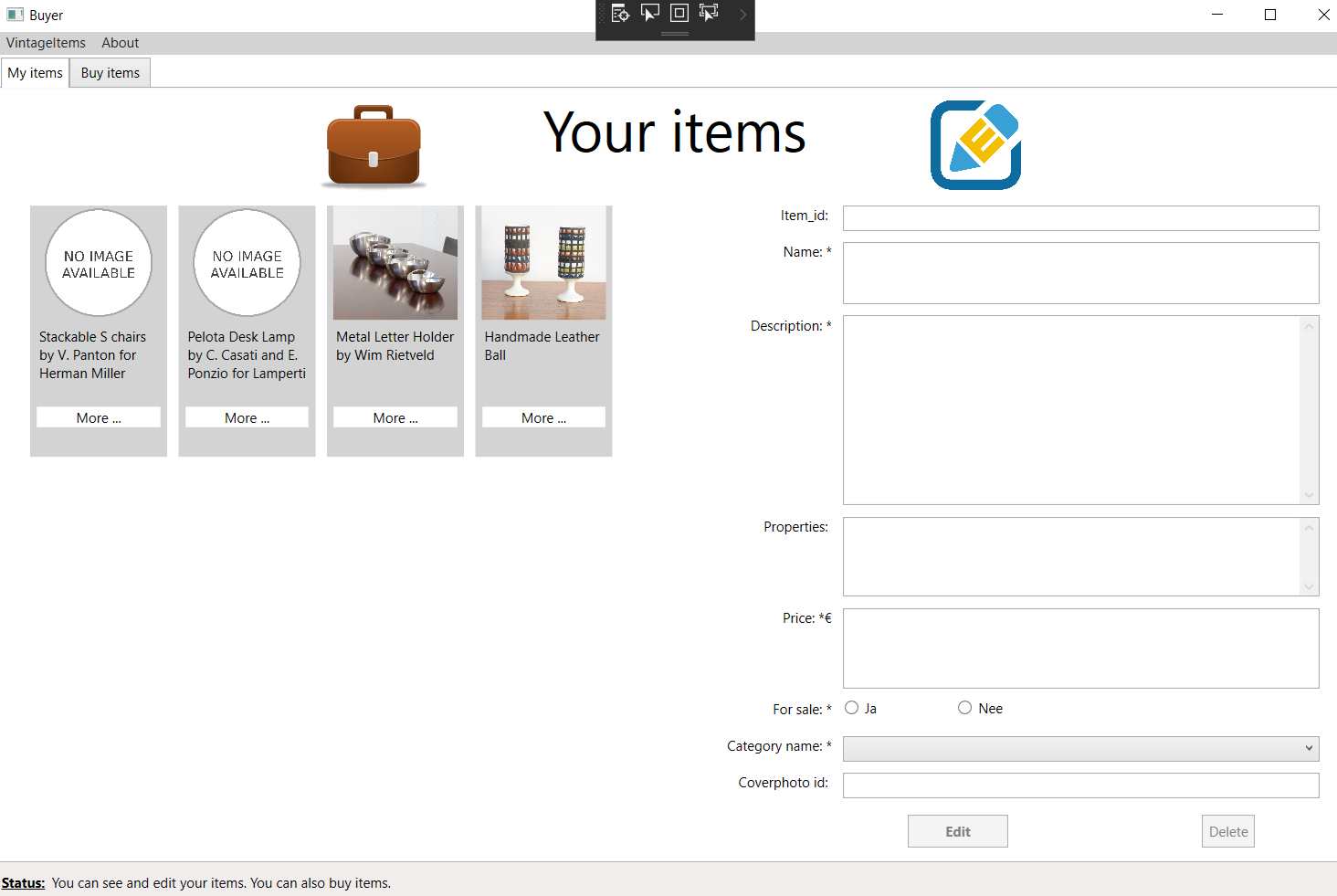


#### Succesvolle login

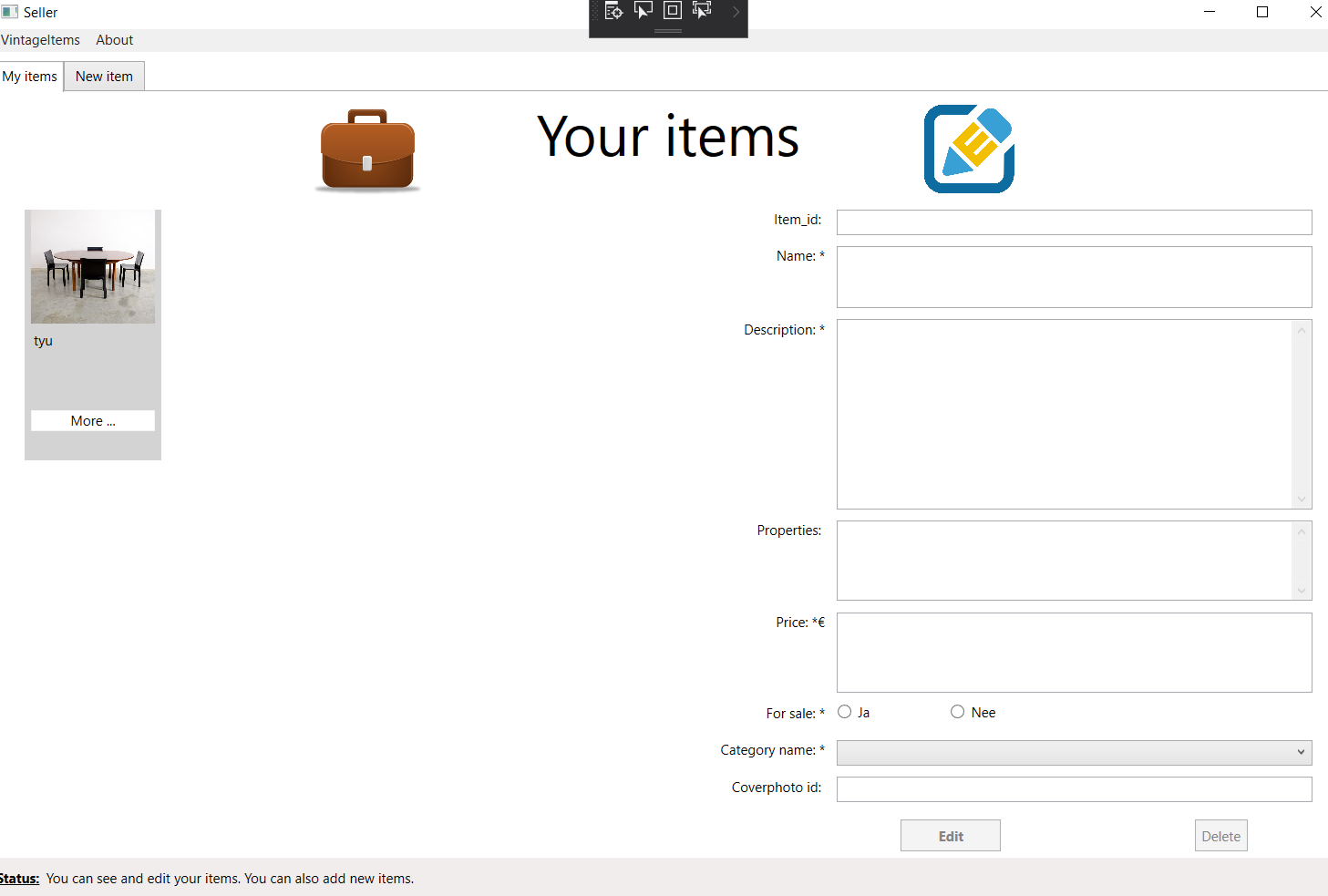
##### Admin



##### Buyer



##### Seller



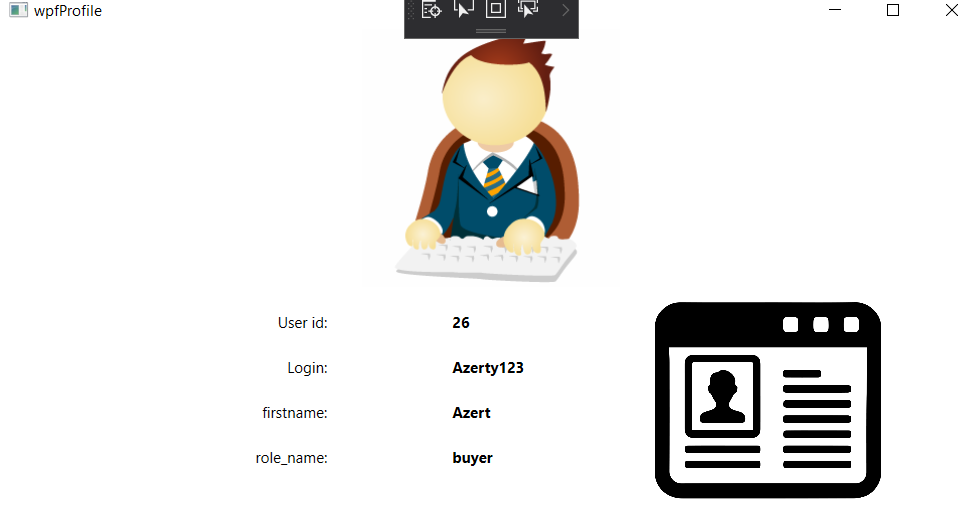
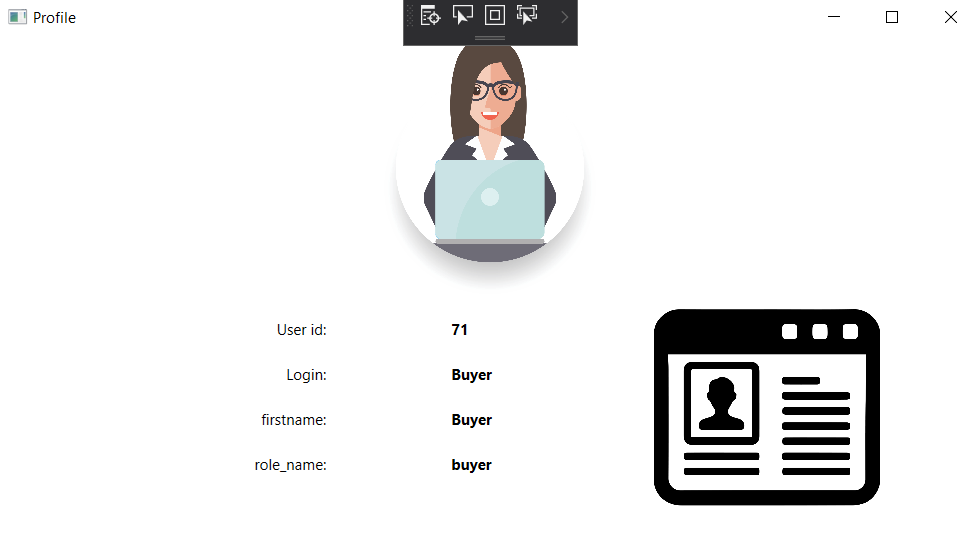
## Algemene toepassingen bij alle rollen

### Menubalk

Elke project is voorzien van een menubalk en een tab-menu. In de menubalk bevinden er zich onder het item “VintageItems” twee subitems.

#### wpfProfile

Wanneer er op profile geklikt wordt zal er informatie weergegeven worden van de gebruiker in een nieuwe wpf. Afhankelijk van het geslacht van de user zal het icoontje een man of een vrouw zijn.

#### wpfAbout

Bij het klikken op “About” zal de wpfAbout tevoorschijn komen. Hier wordt info weergegeven over de applicatie.

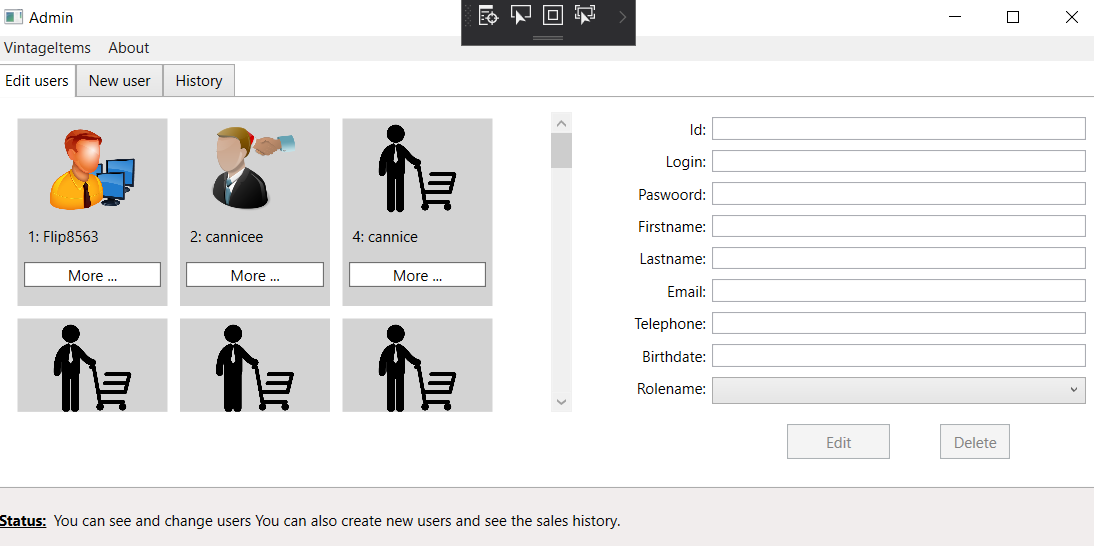


#### WpfLogout

Wanneer er op “Log out” geklikt wordt, wordt de applicatie volledig afgesloten. Indien je opnieuw wenst in te loggen moet je de wpfLogin opnieuw opstarten.

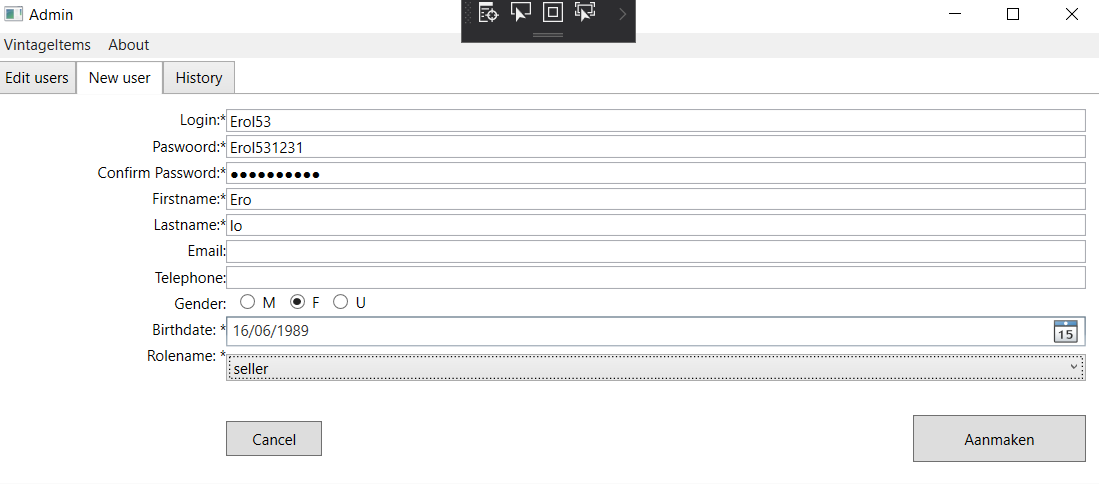
## wpfAdmin

#### Edit users

Op dit tabitem worden alle users weergegeven. Afhankelijk van de rol (User, Seller of Buyer) zal er een aangepast icoontje verschijnen. 

Wanneer je op de “More” button klikt, zal de info weergegeven worden in de labels. Je kan wijzigingen aanbrengen en deze opslaan door op “Edit” te klikken. Je kan de aangeklikte user ook verwijderen door op “Delete” te klikken.

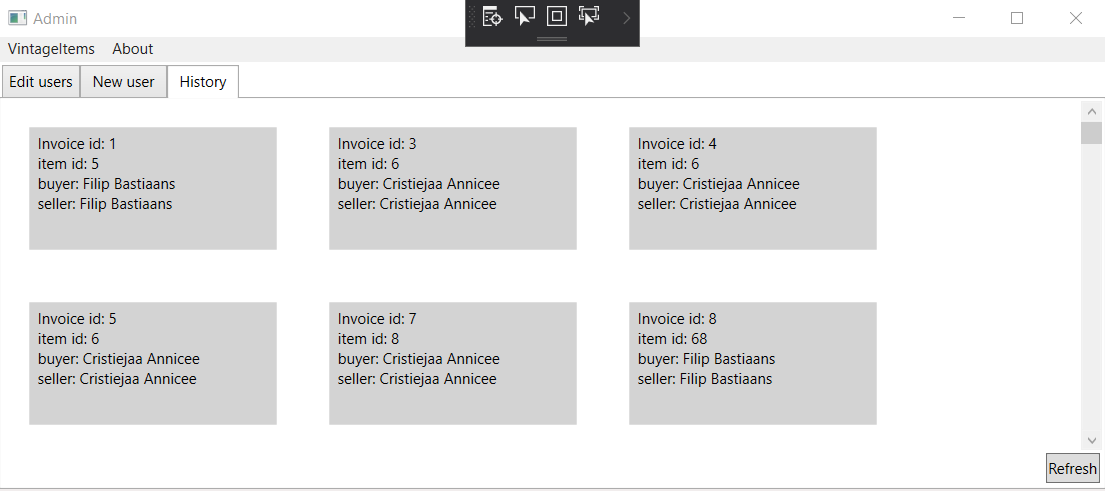
#### New user

Hier create je een nieuwe user. Je voert alle nodige gegevens in en vervolgens klik e op create.  Als alles goed is ingevuld zal de user gecreate worden.



#### History

Hierop zie je alle items die verkocht zijn in de vorm van facturen. Je kan deze enkel bekijken en op geen enkele manier wijzigen.

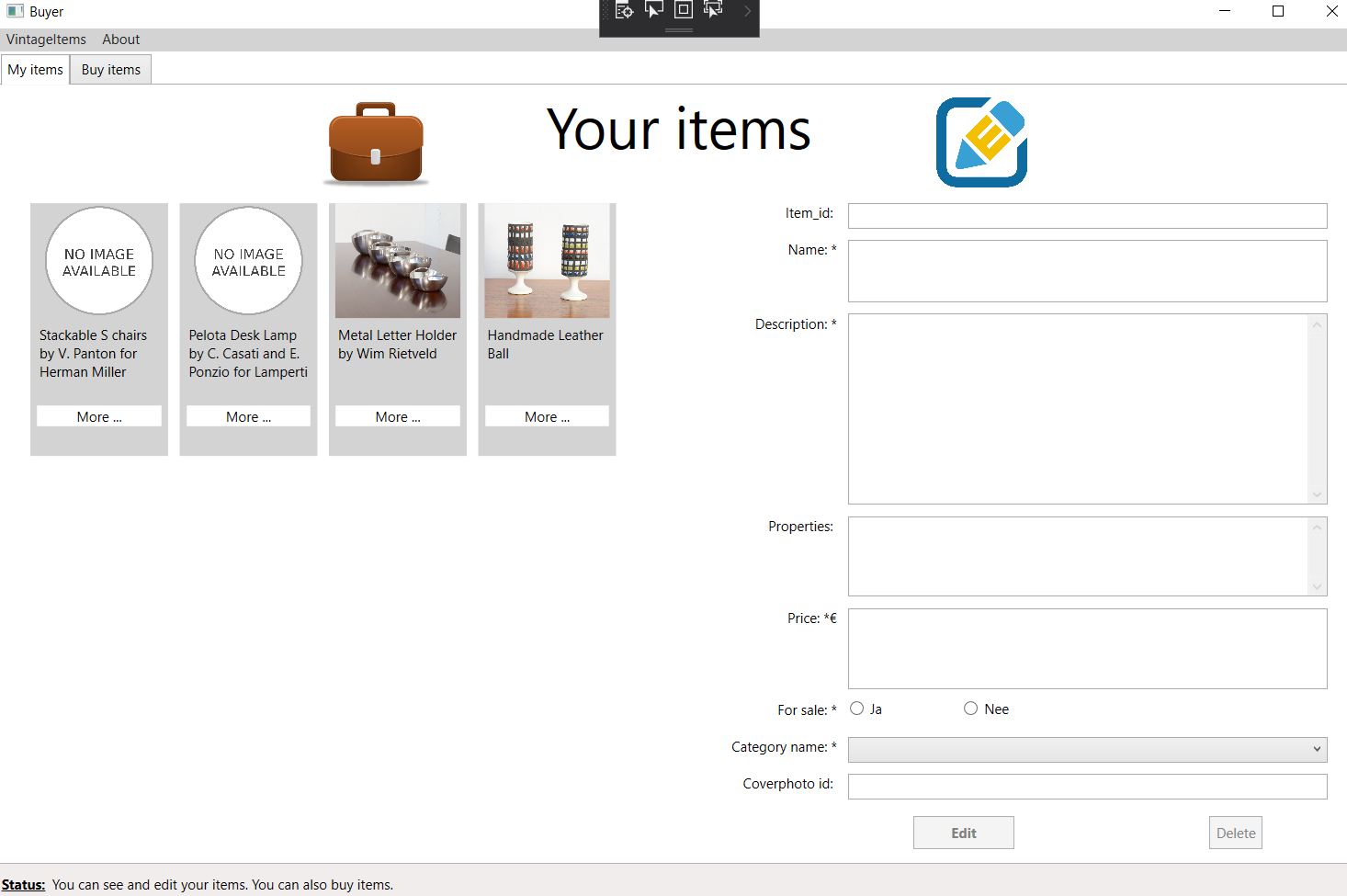


Er is ook een refresh button. Dit kan handig zijn als de buyer net een item heeft gekocht en er iets mis loopt. Hij zal de admin contacteren en om er zeker van te zijn dat deze toegevoegd wordt in de lijst bezit dit scherm over een refresh knop.

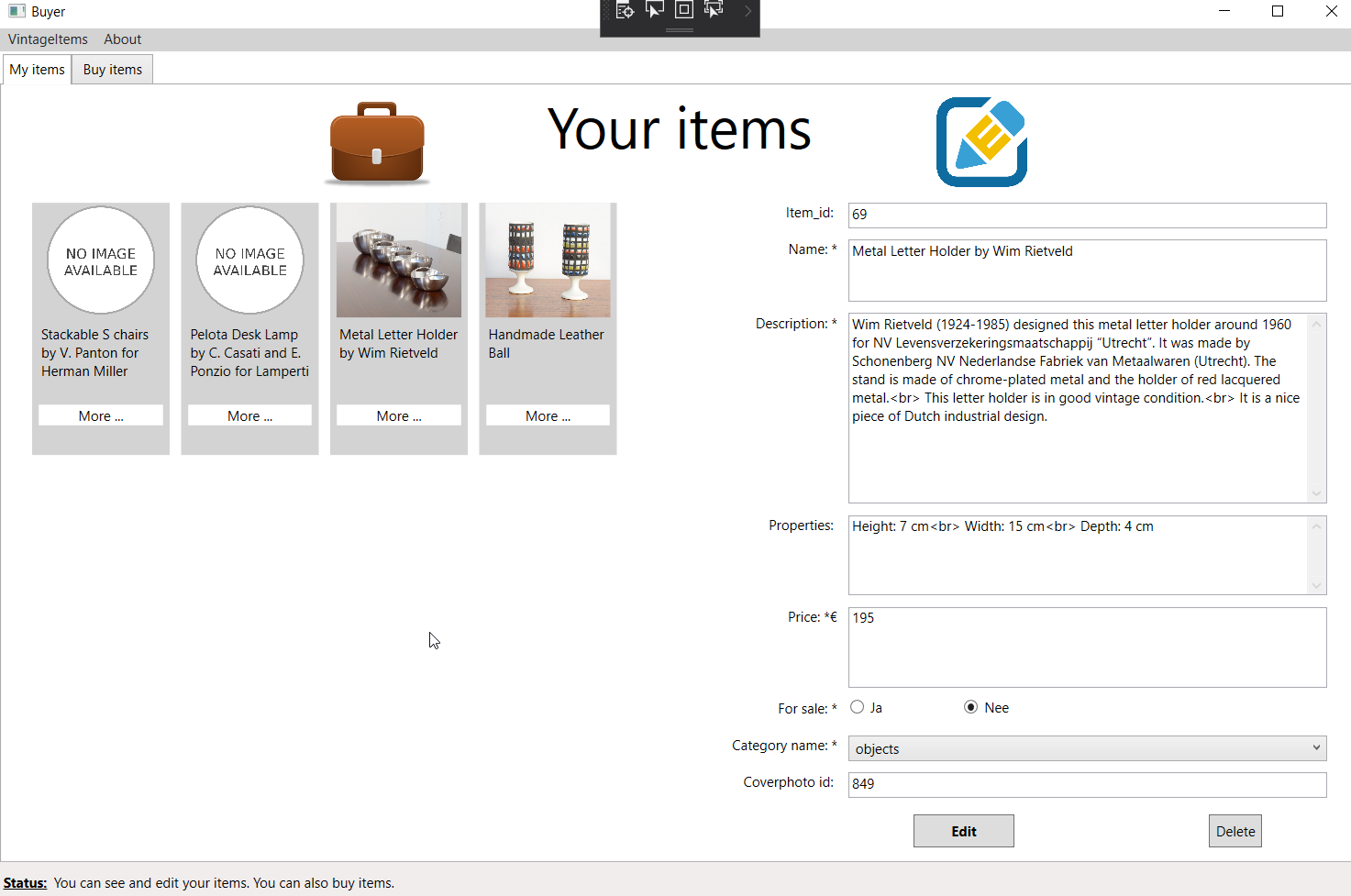
## wpfBuyer

#### My items

Wanneer de user een buyer is, krijgt hij zijn items weergegeven onder het tabitem “My items”.



Wanneer er op de “More” knop geklikt wordt, wordt de info van dat item rechts weergegeven in de labels.



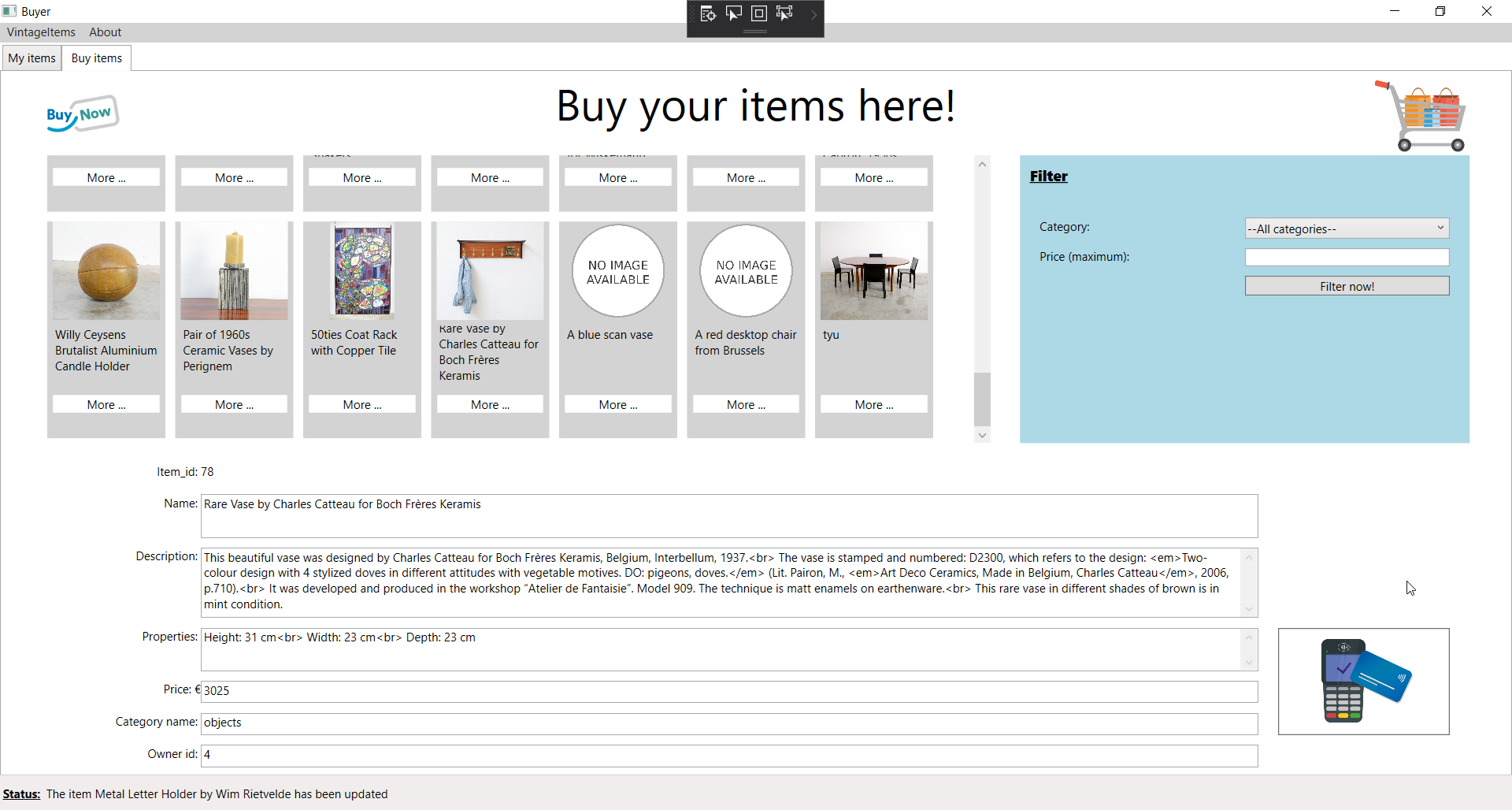
Als de velden het toestaan kan je wijzigingen aanbrengen. Om deze op te slaan moet je klikken op “Edit”. Er zal een bevestigingsmeldiging komen als het wijzigen gelukt is.



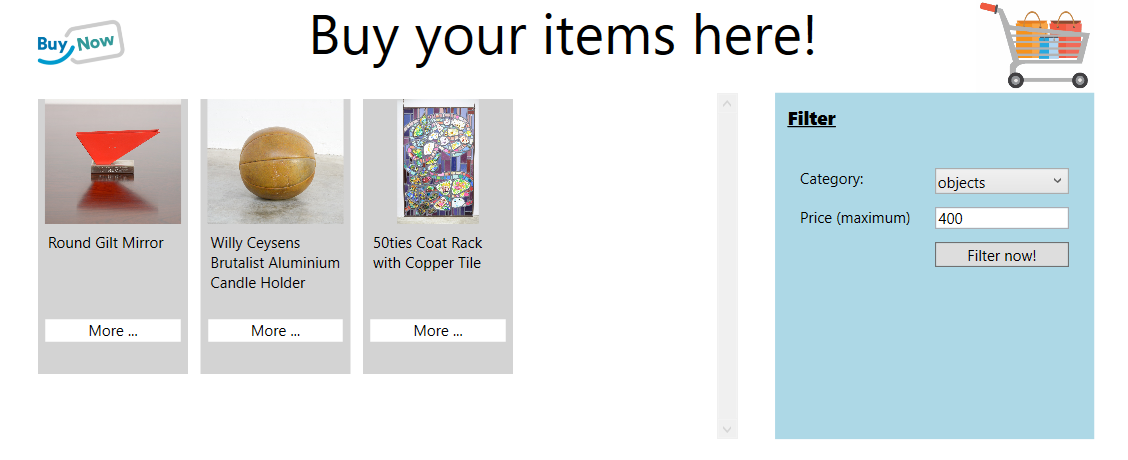
Wanneer je op “Delete” klikt zal het item verwijderd worden uit jouw items. Hij zal ook uit de database verwijderd worden.

#### Buy items

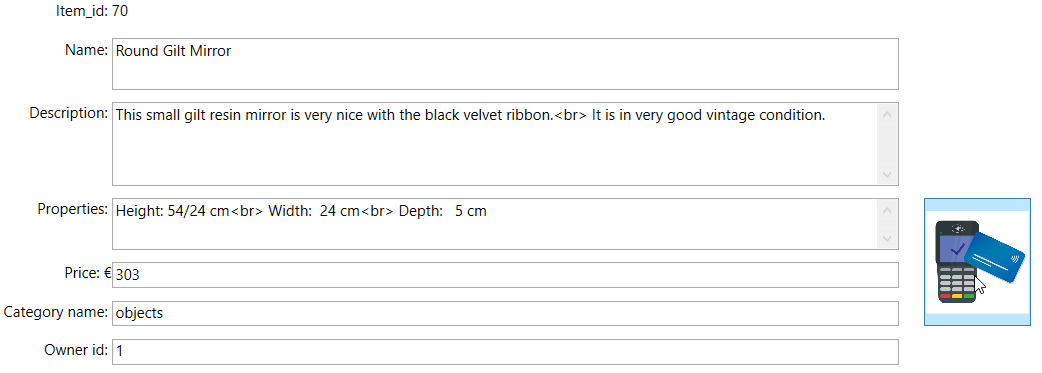
Je kan als buyer kopen. Op het tabitem “Buy items” zullen alle te koop gestelde items weergegeven worden. wanneer op de “More” button klikt van het item zal de nodige info ingevuld worden.



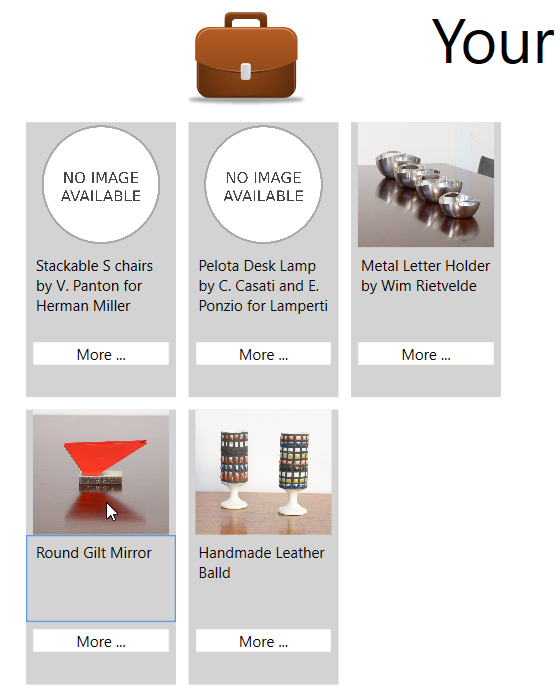
Je kan een filter gebruiken om de weergegeven items aan te passen.



Tot slot kan je nog een item kopen. Hiervoor moet je op de “More” button van het item klikken. Vervolgens klik je op de betaalmachine en zal het item je eigendom worden.



Je zal zien dat het gekochte item bij je items zal komen.

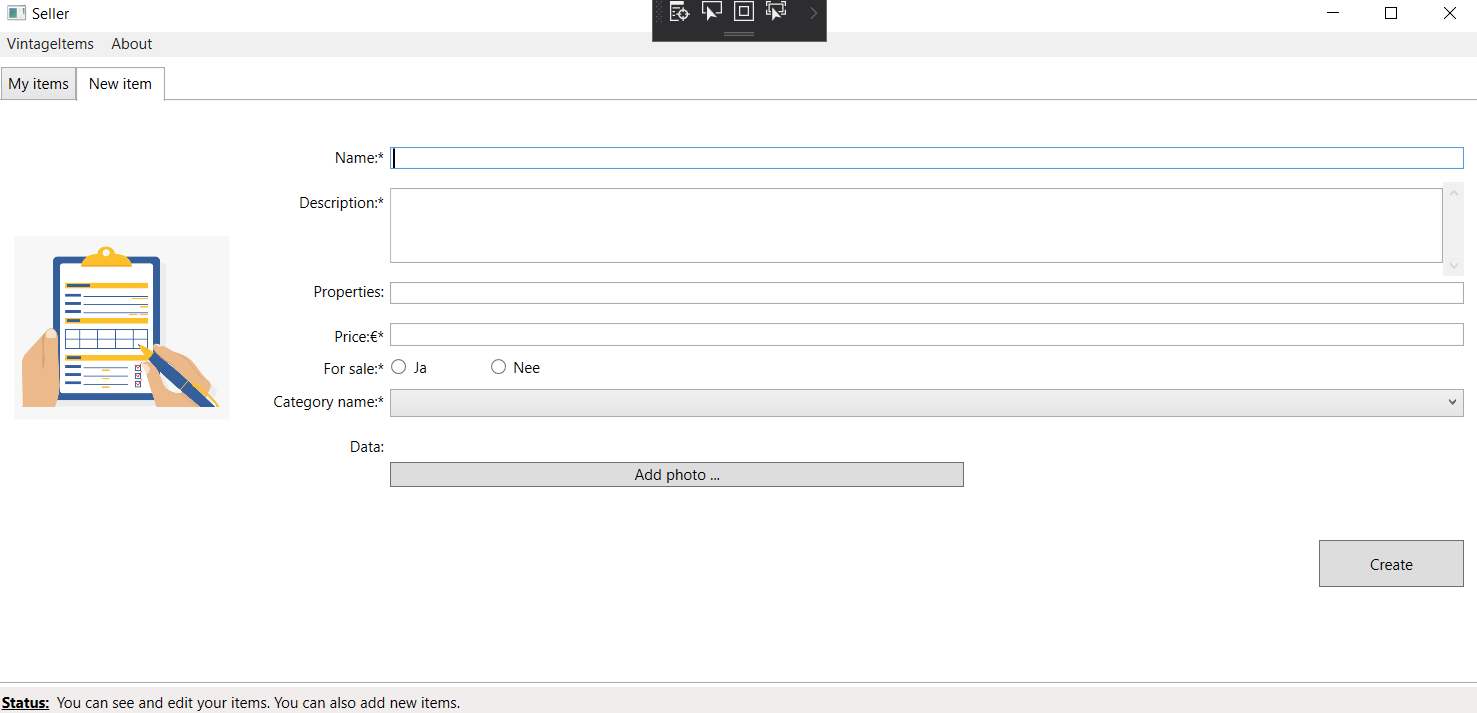


## wpfSeller

#### My items

Deze werkt identiek als “My items” van de buyer. Voor de werking zie wpfBuyer:My items.

#### New item



Je vult de nodige gegevens in van het item. Wanneer je klikt op “Create” zal het programma controleren of alle velden die aangegeven zijn met “\*” weldegelijk ingevuld zijn. Je mag maar moet geen afbeelding en/of properties toevoegen. Als alles ingevuld is zal het item gecreëerd worden en toevoegt worden aan de list met je items.

