ASP.NET Project – Pawn Shop – Заложна къща

## Overview

Web базирана backend система за заложни вещи.

Потребителите, трябва да се регистрират и да станат Клиент, за да ползват услугите на заложната къща.

**Клиентите** сключват договори за заем или продажба на вещи/стоки. Договорите могат да се сключват от Клиентите. Клиентите могат да виждат само собствените си договори, които могат да редактират докато статуса на договора е „Awaiting approval“. Клиентите могат да внасят лихва само по собствените си договори, които са активни към момента, т.е статусът на договора е „Approved“ или „Late“.

**Администратора** може да вижда всички договори и да ги редактира.

**Всички (вкл. нерегистрирани потребителите)** могат да виждат активните стоки в магазин и да ги купуват.

Само **Администратора** може да редактира данните на стоките в магазин.

**Договорът** може да бъде:

* за продажба в магазин
* за залог на вещи срещу сума пари

**Срок** на Договора - може да бъде сключен за:

* 0 дни (стоката отива директно в магазин, а статусът на договора се променя на (For a shop)
* 1 - 30 дни - дължи лихва по 30% на ден

**Статусът** на **Договор** може да бъде:

* Awaiting approval
* Approved ( Active )
* Finished (приключен - вещите са откупени или са за продажба в магазин)
* Late
* For a shop (вещите/стоките отиват в магазин от Администратор)

**Договорите** могат да бъдат редактирани и изтривани (soft deleted) само от Администратор.

**Магазин** - стоките/вещите в **магазин**, могат да бъдат:

* NotSold ( докато SoldDate = null)
* Sold (когато (SoldDate != null)

**Стоките** в магазин могат да бъдат редактирани и изтривани (soft deleted) само от Администратор.

**Лихвите** по договорите, могат да:

* се внасят от Клиента по договора, което удължава срока на договора с броя дни, за които е сключен
* да се изтриват/анулират от Администратора при грешно въвеждане или неполучена сума

**Клиентите** могат да бъдат изтривани от Администратор, само когато нямат нито един сключен договор.

Договори могат да бъда изтривани от Администратор, само когато клиентът се отказва.

## Identity Requirements

Използва се **default** **IdentityUser с**ъс следните изисквания за парола:

The password requirements for the IdentityUser are the following:

* Require confirmed account: **false**
* Require digits: **false**
* Require non-alphanumeric characters: **false**
* Required uppercase letters: **false**

## Database Requirements

The **Database** of **PawnShopDB**:

**Client**

* Has **Id –** a unique int  **, Primary Key (по-добре е да е Giud, но за по-бързо го правя с int)**
* **Has PhoneNumber – string (required)** min/max length **10(required)**
* **Has Address – string required**
* **Has UserId – string (required)(foreignKey)**
* **Has User – IdentityUser/ApplicationUser**
* **Has Collection of Agreement**
* **Has IsDeleted – bool – default = false**

**Agreement**

* Has **Id –** a unique  **int, Primary Key (по-добре е да е Giud, но за по-бързо го правя с int)**
* Has **GoodName** – **string** with min length **3** and max length **50(required)**
* **Has Description – string (optional)**
* Has **Price – decimal (required) min 20 – max 1000 (required)**
* Has **ReturnPrice – decimal (required) min 20 – max 1000 (required)**
* Has Duration **– integer (required)**
* Has **Ainterest** – **decimal/ not required**
* Has **StartDate** – **DateTime** with format "**dd/MM/yyyy**" (**required**)
* Has **EndDate** – **DateTime** with format "**dd/MM/yyyy**" (**required**)
* Has **UserId** - **string**
* Has **ApplicationUser** Account
* **Has IsDeleted – bool – default = false**
* **Has AgrreementStateId-** **int (required) (foreign key)**
* Has **AgrreementState**– AgrreementState**(required)**
* **Has Collection Interests, type of Interest**

**Interest**

* Has **Id –** a unique  **integer, Primary Key**
* Has **AgreementId – int (required) foreign key (required)**
* **Has Agreement - Agreement**
* Has **ValueInterest – decimal (required) min 1 – max 1000 (required)**
* Has **DateInterest** – **DateTime** with format "**dd/MM/yyyy**" (**required**)
* Has **UserId** - **string**
* Has **ApplicationUser** Account
* **Has IsDeleted – bool – default = false**

**Shop**

* Has **Id –** a unique  **integer, Primary Key**
* Has **Name** – **string**
* **Has Description – string (optional)**
* Has **AgreementId – int (required) foreign key (required)**
* **Has Agreement - Agreement**
* Has **SellPrice – decimal (required) min 1 – max 100000**
* Has **SoldDate** – **DateTime** with format "**dd/MM/yyyy**" optional
* **Has IsDeleted – bool – default = false**

**ContractState**:

* Has **Id –** a unique  **integer, Primary Key**
* **Has Name string** with min length **3** and max length **30(required)**
* Аwaiting approval
* Approved (Active)
* Finish
* Late
* For a Shop

**Connection string default:**

"Server=.\\SQLEXPRESS; atabase=PawnShop;Trusted\_Connection=True;TrustServerCertificate=True;"

## SEED DATA

private void SeedUsers()

{

var hasher = new PasswordHasher<IdentityUser>();

AdminUser = new ApplicationUser()

{

Id = "b97f29e7-1edd-4666-a8d8-8882858d7ccf",

UserName = "admin@abv.bg",

NormalizedUserName = "ADMIN@ABV.BG",

Email = "admin@abv.bg",

NormalizedEmail = "ADMIN@ABV.BG",

FirstName = "Admin",

LastName = "Boss"

};

AdminUser.PasswordHash = hasher.HashPassword(AdminUser, "123456");

UserMsef = new ApplicationUser()

{

Id = "5cf194c7-b26e-42ed-8efd-d98c7980373b",

UserName = "msef@abv.bg",

NormalizedUserName = "MSEF@ABV.BG",

Email = "msef@abv.bg",

NormalizedEmail = "MSEF@ABV.BG",

FirstName = "Mary",

LastName = "Seferov"

};

UserMsef.PasswordHash = hasher.HashPassword(UserMsef, "123456");

UserKsef = new ApplicationUser()

{

Id = "ffae7662-4ff3-4698-8f36-c4e4f392da18",

UserName = "ksef@abv.bg",

NormalizedUserName = "KSEF@ABV.BG",

Email = "ksef@abv.bg",

NormalizedEmail = "KSEF@ABV.BG",

FirstName = "Kalin",

LastName = "Sarafov"

};

UserKsef.PasswordHash = hasher.HashPassword(UserKsef, "123456");

GuestUser = new ApplicationUser()

{

Id = "70e39283-bd42-4d0a-aa2e-46d2a31c4f87",

UserName = "guest.abv.bg",

NormalizedUserName = "GUEST@ABV.BG",

Email = "guest@abv.bg",

NormalizedEmail = "GUEST@ABV.BG",

FirstName = "Guest",

LastName = "Guestov"

};

GuestUser.PasswordHash = hasher.HashPassword(GuestUser, "123456");

}

private void SeedClient()

{

ClientMsef = new Client

{

Id = 1,

PhoneNumber = "1234567890",

Address = "Sofiq,Dianabad",

UserId = UserMsef.Id,

IsDeleted = false

};

ClientKsef = new Client

{

Id = 2,

PhoneNumber = "1234599999",

Address = "Plovdiv,Izgrev",

UserId = UserKsef.Id,

IsDeleted = false

};

AdminClient = new Client

{

Id = 3,

PhoneNumber = "8888899999",

Address = "Varna,Center",

UserId = AdminUser.Id,

IsDeleted = false

};

}

private void SeedState()

{

AwaitingApproval = new AgreementState

{

Id = 1,

Name = "Аwaiting approval"

};

Approved = new AgreementState

{

Id = 2,

Name = "Approved (Active)"

};

Finished = new AgreementState

{

Id = 3,

Name = "Finished"

};

Late = new AgreementState

{

Id = 4,

Name = "Late"

};

ForShop = new AgreementState

{

Id = 5,

Name = "For а Shop"

};

}

private void SeedAgreement()

{

Laptop = new Agreement

{

Id = 1,

GoodName = "лаптоп Acer",

Description = "4GB RAM, 120SSD",

Price = 200,

ReturnPrice = 201,

Duration = 10,

StartDate = new DateTime(2024, 11, 25, 0, 0, 0),

EndDate = new DateTime(2024, 12, 5, 0, 0, 0),

UserId = UserKsef.Id,

AgrreementStateId = 1,

Ainterest = 1

};

TV = new Agreement

{

Id = 2,

GoodName = "SONY TV",

Description ="used - produced 2012",

Price = 100,

ReturnPrice = 103,

Duration = 30,

StartDate = new DateTime(2024, 11, 26, 0, 0, 0),

EndDate = new DateTime(2024, 12, 26, 0, 0, 0),

UserId = UserMsef.Id,

AgrreementStateId = 2,

Ainterest = 3

};

Bike = new Agreement

{

Id = 3,

GoodName = "Bike",

Description = "model Balkan",

Price = 30,

ReturnPrice = 33,

Duration = 30,

StartDate = new DateTime(2024, 11, 27, 0, 0, 0),

EndDate = new DateTime(2024, 12, 27, 0, 0, 0),

UserId = UserMsef.Id,

AgrreementStateId =5

};

}

private void SeedShop()

{

BikeForShop = new Shop

{

Id = 1,

Name = Bike.GoodName,

AgreementId = Bike.Id,

SellPrice = Bike.Price

};

}

private void SeedInterest()

{

TVInterest = new Interest

{

Id = 1,

ValueInterest = TV.Ainterest,

DateInterest = DateTime.Now,

AgreementId = TV.Id,

UserId = TV.UserId,

};