Apartman Yönetimi Sistemi (Apartment Management System)

Hazırlayan: Zehra Göl

Giriş

Bu çalışmada bir apartman yönetim sistemi gerçekleştirilmiştir. Bu sistem bir apartmanın yönetiminde kullanılabilecek, faturalarla, dairelerle veya apartmanda oturan kişilerle ilgili çeşitli fonksiyonları bir API aracılığıyla apartman yöneticisinin ve apartman sakinlerinin kullanımına sunmaktadır.

Bu sistemin yönettiği 4 farklı varlık (**entity**) bulunmaktadır;

- Apartman (**MainBuildings**)
- Daireler (**Flats**)
- Kullanıcılar ve yönetici (**Users**)
- Fatura ve ödemeler (**Payments**)

Bu dört varlığın (entity) fonksiyonları farklı yetkilendirmeler ile (**authentication**) iki farklı tür kullanıcı tarafından kullanılmaktadır;

- Daire sahibi veya kiracı (**Kullanıcı** rolü ile)
- Yönetici (**Admin** rolü ile)

Kullanıcı, daire, apartman ve ödemeler ile ilgili bilgiler ve gerçekleştirilen işlemler ile ilgili kayıtlar **Microsoft SQL Server** içinde çeşitli tablolarda, çeşitli ilişki türleri **(bire-bir, bire-çok)** ile tutulmaktadır.

Raporun sonraki kısımlarında;

1. Veritabanındaki Tablolar ve Özellikleri
2. Tabloların İlişki Türleri
3. Sistemdeki Roller ve Görevleri
4. Uygulamanın Dosya Düzeni
5. Sistemin Başlangıç Fonksiyonları
6. Akışlar ve Kullanım Senaryoları **(Use Case)** Diyagramları
7. Daire (**Flat**), Apartman (**MainBuilding**), Kullanıcı (**User**), Ödeme (**Payment**) Varlıklarının **Fonksiyonlarının** Tanıtımı

sırasıyla açıklanacaktır.

Teşekkür

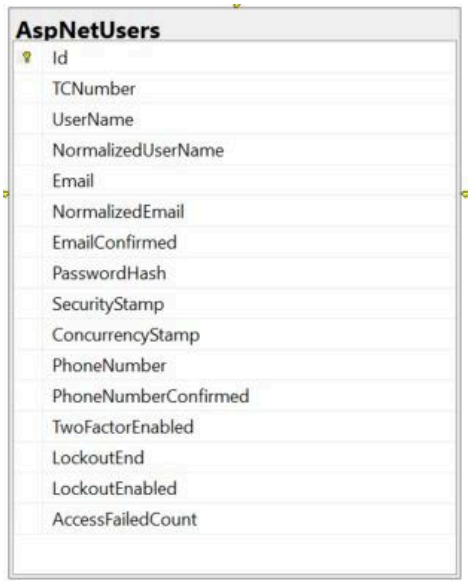
Bu eğitimi gerçekleştiren papara'ya ve değerli ekibine, dersleri anlatan Sayın ***Fatih Çakıroğlu***'na ve programın koordinatörlüğünü üstlenen **Coderspace** ekibine sonsuz teşekkürlerimle.

Veritabanı Tanıtımı

Bu kısımda veritabanındaki tabloların tanımı ve varlıkların (entity) özelliklerinden bahsedilecektir. Kullanıcı, daire ve ödeme bilgilerini içeren tablolar sırasıyla;

Kullanıcı Tablosu (AspNetUsers)

Bu varlık (entity) **Identity Framework** kullanılarak oluşturulmuştur.



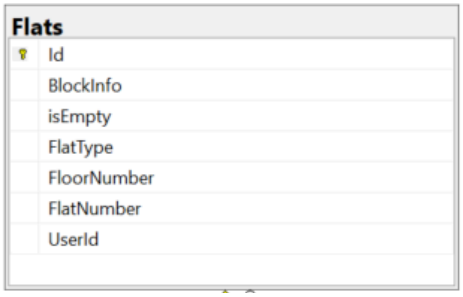
Tablo Alanları:

- Kullanıcı Adı (**UserName**)
- TC Kimlik Numarası (**TCNumber**)
- Telefon Numarası (**PhoneNumber**)
- Şifre (**Password**) (sadece yönetici için)

Navigation Properties:

- Daire Listesi (**FlatList**)
- Ödeme Listesi (**PaymentList**)
- Apartman (**Main Building**)

Daire Tablosu (Flats)



Tablo Alanları:

- Blok Bilgisi (**BlockInfo**)
- Doluluk Bilgisi (**isEmpty**)
- Daire Tipi (**FlatType**) Örn: “2+1”, “3+1”
- Kat Numarası (**FloorNumber**)
- Daire Numarası (**FlatNumber**)

Foreign Key:

- Kullanıcı ID’si (**UserId**)

Apartman Tablosu (MainBuildings)

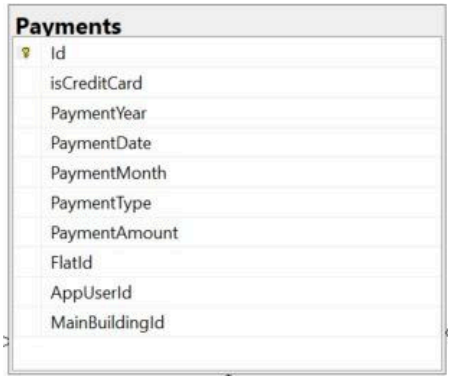
Tablo Alanları:

- Apartmanın İsmi (**Name**) Örn: “Barış Apt.”
- Bina Yaşı (**BuildingAge**) Örn: “30+”

Foreign Key:

- Kullanıcı ID’si (**UserId**)

Ödeme Tablosu (Payments)



Tablo Alanları:

- Ödeme Yöntemi (**isCreditCard**)
- Fatura Yılı (**PaymentYear**)
- Fatura Ayı (**PaymentMonth**)
- Ödendiği Tarih (**PaymentDate**)
- Fatura Tipi (**PaymentType**) Örn: “Aidat”, “Su”
- Fatura Miktarı (**PaymentAmount**)

Foreign Key:

- Daire ID’si (**FlatId**)
- User ID’si (**AppUserId**)
- Apartman ID’si (**MainBuildingId**)

Varlıkların İlişkileri (Relationships)

Bu çalışmada 3 tane bire çok, 1 tane bire bir ilişki bulunmaktadır. Bunlar sırasıyla;

- Kullanıcı (**User**) - Daire (**Flat**) → bire çok ilişki
  Bir kullanıcın birden çok dairesi olabilir fakat bir dairenin sadece bir kullanıcı olabilmektedir.

- Kullanıcı (**User**) - Ödeme (**Payment**) → bire çok ilişki
  Bir kullanıcının birden çok ödemesi olabilir ama bir ödemenin sadece bir kullanıcısı olabilmektedir.

- Apartman (**MainBuilding**) - Ödeme (**Payment**) → bire çok ilişki
  Bir apartmanın birden çok ödemesi olabilir fakat bir ödemenin sadece bir apartmanı olabilir.

- Daire (**Flat**) - Ödeme (**Payment**) → bire çok ilişki
  Bir dairenin birden çok ödemesi olabilir fakat bir ödeme sadece bir daireye ait olabilmektedir.

- Kullanıcı (**User**) - Apartman (**MainBuilding**) → bire bir ilişki
  Bir apartmanın sadece bir yöneticisi olabilmekte ve bir yöneticinin sadece bir apartmanı olabilmektedir.

Aşağıdaki veritabanı şemasında bu ilişkiler görülebilmektedir.

Not:
Projede istenen varlıklar dışında bir **apartman** varlığı oluşturulmasının sebebi, senaryo kurgulanırken **bir kullanıcının birden fazla daireye** sahip olabileceği şekilde kurgulanmasından dolayı projede bire bir ilişki bulunmamasıdır. **Bire bir** ilişkiyi de elde edebilmek amacıyla ise projeye apartmanı temsil eden bir varlık daha eklenmiş ve kullanıcı ile apartman arasında bire bir ilişki oluşturulmuştur.

Sistemdeki Roller (Authentication) ve Görevleri (Authorization)

Sistemde **Kullanıcı** ve **Yönetici** olarak iki adet rol bulunmaktadır. Bu rollerin güvenlik seviyeleri farklı olup, yöneticinin gerçekleştirdiği işlemleri kullanıcılar gerçekleştirememektedir. Rollerin kullandıkları fonksiyonlar raporun ilerleyen kısımlarında açıklanacaktır, bu kısımda rollerin neler yapabildikleri özetlenmiştir.

Yönetici
Rol Adı: “admin”

- Kullanıcıların bilgilerini; oluşturma **(create)**, güncelleme (**update**) ve silme (**delete**) işlemleri gerçekleştirebilmektedir.
- Daire oluşturabilmekte ve dairelere kullanıcı atayabilmektedir.
- Dairelerin ödemesi gereken aidat, elektrik, su veya doğalgaz gibi **ödemeleri** sistem üzerinden dairelere atayabilmektedir.
- Dairelerin ödemesi gereken aylık aidatı **toplu olarak** dairelere ödeme oluşturarak atayabilmektedir.
- Apartman olarak ödenmesi gereken fatura bilgilerini (elektrik, su, doğalgaz, aidat) aylık olarak **apartman varlığına** (entity) **ödeme** oluşturarak girebilir.

- Dairelerin yapmış olduğu ödemeleri ödenmiş veya ödenmemiş faturaları filtreleyerek görebilmektedir.
- Aylık ve yıllık olarak dairelerin veya apartmanın ödemesi gereken borç durumunu toplam olarak görebilmektedir.
- **BONUS Gerçekleştirildi:** Verilen ödeme tipindeki (aidat, su, elektrik vs.) ödemelerini **düzenli** olarak yapan kullanıcıları görebilmektedir.
- Başlangıçta oluşturulan kullanıcı adını (**"admin"**) ve şifresini (**"Admin12*"**) kullanarak kendisine **token** oluşturabilmektedir.

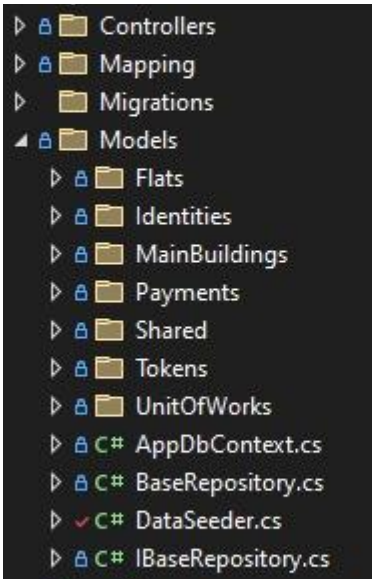
Kullanıcılar
Rol Adı: "kullanici"

- Kendisine atanan fatura ve aidat ödemelerini görüntüleyebilmektedir.
- Kendisine atanan fatura ve aidat ödemelerini ödeyebilmektedir.
- **TC Kimlik** numarasını ve **telefon** numarasını kullanarak kendisine **token** oluşturabilmektedir.

Kullanıcı ödeme yapmak istediğinde;

- Eğer ödemesini vaktinde gerçekleştirmedi ise (aynı ay içinde) **%10 daha fazla** fatura ödemektedir. **(BONUS Gerçekleştirildi)**
- Eğer **geçen yıl** boyunca **"aidat"** ödemesini **düzenli** (aynı ay içinde) ödediyse **bu yıl** gerçekleştireceği "aidat" ödemelerini **%10 daha az** ödeyecektir. **(BONUS Gerçekleştirildi)**

Uygulamanın Dosya Düzeni



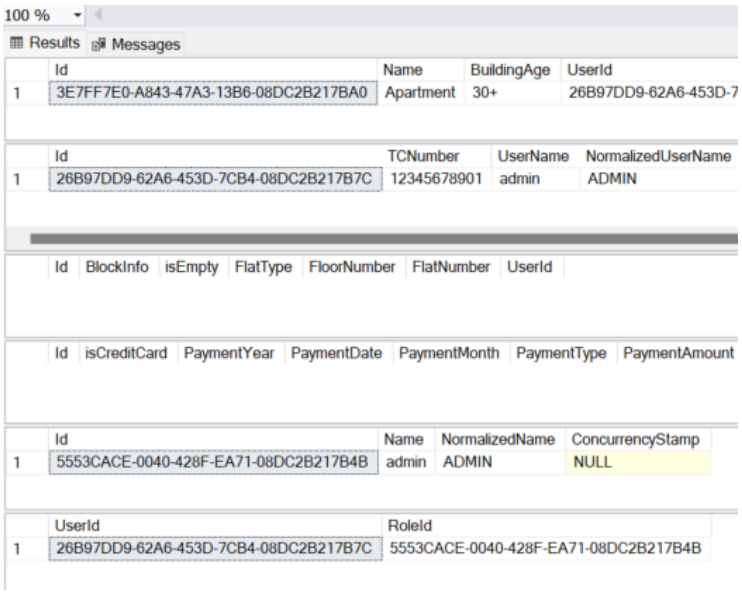
- **Controllers:** Sistemdeki varlıkların ayrı ayrı dört adet controller dosyası burada bulunmaktadır.
- **Mapping:** Mapping işlemi için kullanılan dosya burada bulunmaktadır.
- **Migrations:** Veritabanı güncellemelerinin kayıtları burada bulunmaktadır.
- **Models:**
    - **Flats:** Daire ile ilgili bütün DTO ve servisler bulunmaktadır.
    - **Identities:** Kullanıcı ile ilgili bütün DTO ve servisler bulunmaktadır.
    - **MainBuildings**: Apartman ile ilgili bütün DTO ve servisler bulunmaktadır.
    - **Payments:** Ödemeler ile ilgili bütün DTO ve servisler bulunmaktadır.
    - **Shared:** Bütün varlıkların ortak kullandığı servisler bulunmaktadır.

- **Tokens:** Token işlemleri ile ilgili bütün DTO ve servisler bulunmaktadır.
- **UnitOfWorks:** Ortak olarak kullanılan UnitOfWork servisi dosyaları bulunmaktadır.
- **AppDbContext:** Veritabanına erişim ve manipülasyon işlemleri için kullanılır.
- **BaseRepository:** Bütün varlıkların repository dosyaları buradan inherit almaktadır.
- **DataSeeder:** Program çalışmaya başladığında admin kullanıcısını kullanıcı adı (“admin”) ve şifresiyle (“Admin12*”) birlikte oluşturan ve MainBuilding varlığından bir apartman oluşturan kod burada bulunmaktadır.

Sistemin Başlangıç Fonksiyonları ve Akış Diyagramı

Sistem çalışmaya başladığında **DataSeeder** içindeki iki fonksiyon çalışarak sırasıyla:

1. “admin” adında bir **rol** oluşturur.
2. Kullanıcı adı, şifresi ve diğer bilgileri dahil olmak üzere bir **admin kullanıcısı** oluşturup veritabanına yazar.
3. MainBuilding tipinde bütün bilgileri dahil olmak üzere bir **apartman** oluşturup veritabanına yazar.
4. Ardından oluşturulan **apartmana admin kullanıcısını atar** ve veritabanına kaydeder.



→ Apartman Tablosu

→ Kullanıcı Tablosu

→ Daire Tablosu

→ Ödeme Tablosu

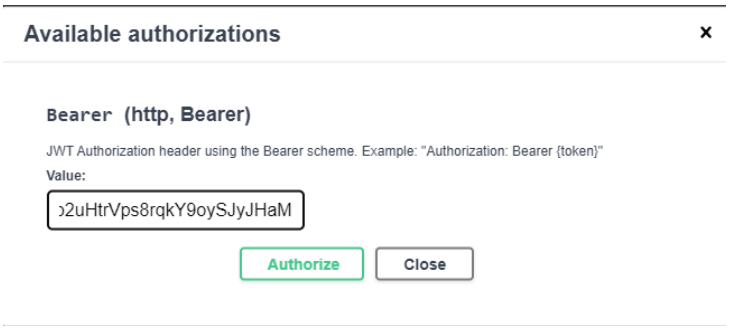
→ Rol Tablosu

→ Kullanıcı-Rol Tablosu

Sistem ayağa kalktığında oluşan tabloların görünüşü bu şekildedir.

Akış Diyagramı (Admin):

1. Admin kullanıcısı **Tokens->CreateTokenForAdmin** fonksiyonunu kullanıcı adı ve şifresini kullanarak kendine bir admin tokeni oluşturur.
2. Ardından **Swagger** veya **Postman** üzerinden kendini yetkilendirir. (Swagger üzerinden de yetkilendirme yapılabilmesi için düzenleme yapılmıştır)



3. Kendini yetkilendiren yönetici sistemdeki bütün fonksiyonları kullanabilmektedir.

Kullanılabilecek fonksiyonlar aşağıda açıklanacaktır.

Akış Diyagramı (Kullanıcı):

Kullanıcı akış diyagramını denemek için aşağıdaki adımlar gerçekleştirilmelidir:

1. “admin” olarak token alınmalı.
   (Tokens->CreateTokenForAdmin)

   ```
   {
     "userName": "admin",
     "password": "Admin12*"
   }
   ```

2. Seçilen bir kullanıcıya aşağıdaki yapı ile ve **tokens/AssignRoleToUser** fonksiyonu ile **“kullanici”** rolu atanmalı. Böyle bir rol yok ise sistem tarafından oluşturulmaktadır.

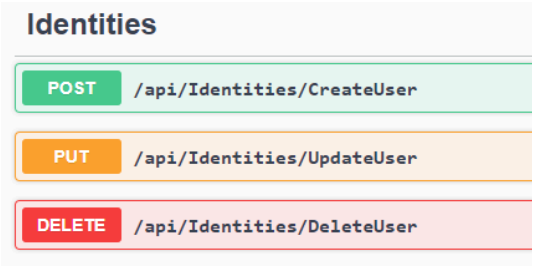
   ```
   {
     "userId": "D66429FF-F24D-4BBD-9044-08DC2B25EF1C",
     "roleName": "kullanici"
   }
   ```

3. Elde edilen **token** ile yetkinlendirme yapılmalı.
4. Ardından kendine atanan token ile fatura ödeme işlemi (**PayBillWithGivenFlatNumber**) veya ödenmiş/ödenmemiş faturalarını görme (**GetPaymentsWithGivenFlatNumber**) işlemi yapabilmektedir.

Fonksiyonların Tanıtımı

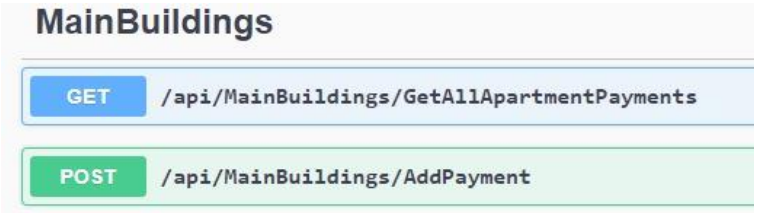
Bu projede gerçekleştirilen bütün fonksiyonlar burada tanıtılacaktır.

Kullanıcı Fonksiyonları



- Buradaki bütün fonksiyonları sadece **admin** rolüne sahip kişi gerçekleştirebilir.
- Kullanıcı **oluşturma** ve **güncelleme** için:
  ```
  {
    "fullName": "ZehraGoll",
    "tcNumber": "13739116234",
    "email": "zehragol@gmail.com",
    "phoneNumber": "5445520453"
  }
  ```
  yapısına uygun girdiler verilip çalıştırılabilir. Kullanıcı **silme** işlemi için ise:
  ```
  {
    "tcNumber": "13739116234"
  }
  ```
  sadece TC numarası verilip kullanıcı silinebilir.

Apartman Fonksiyonları

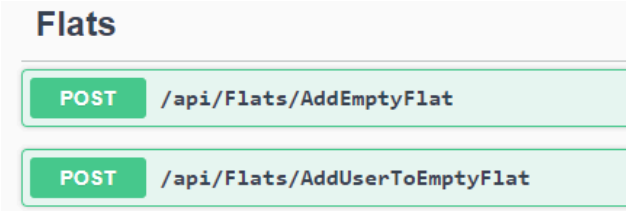


- Buradaki bütün fonksiyonları sadece **admin** rolüne sahip kişi gerçekleştirebilir.

- Apartmana ait bütün faturaları görmek için **GetAllApartmentPayments** fonksiyonu parametre vermeden çalışıtırılabilir.

- Apartmana bir fatura eklemek için aşağıdaki fatura bilgileri girilerek **AddPayment** fonksiyonu çalıştırılabilir.
  ```
  {
    "paymentYear": 2024,
    "paymentMonth": 2,
    "paymentType": "elektrik",
    "paymentAmount": 350
  }
  ```

Daire Fonksiyonları



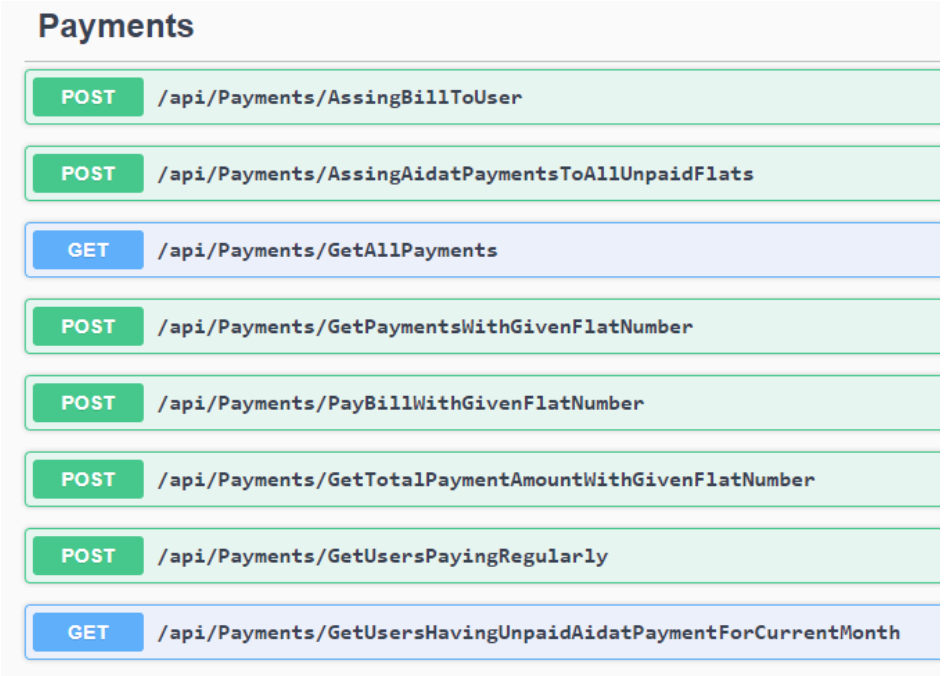
- Buradaki bütün fonksiyonları sadece **admin** rolüne sahip kişi gerçekleştirebilir.
- Dairenin bilgileri aşağıdaki şekilde girilerek **AddEmptyFlat** fonksiyonu ile boş bir daire eklemesi gerçekleştirilebilir.

```
{
  "blockInfo": "A Block",
  "flatType": "3+1",
  "floorNumber": "1",
  "flatNumber": 3
}
```

- Daireye kullanıcı ataması aşağıdaki şekilde girerek **AddUserToEmptyFlat** fonksiyonu ile geçekleştirilebilir.

```
{
  "userTCNumber": "13739116234",
  "flatNumber": 3
}
```

Ödeme Fonksiyonları



Not: PaymentDate yani Ödeme Tarihi kısımlı alan faturanın ödenmiş veya ödenmemiş olma bilgisini de içermektedir. Eğer bu alan boşsa fatura ödenmemiş, dolu ise ödenmiş demektir.

AssignBillToUser

- Bu fonksiyonu sadece **admin** kullanabilir.

```
{
  "paymentYear": 0,
  "paymentMonth": 0,
  "paymentType": "string",
  "paymentAmount": 0,
  "flatNo": 0
}
```

- Yukarıdaki yapıya uyarak bir daireye fatura ataması yapılabilmektedir. Atanan ödeme veritabanındaki **Payments** tablosuna kaydedilir.

AssingAidatPaymentsToAllUnpaidFlats

- Bu fonksiyonu sadece **admin** kullanabilir.

```
{
  "paymentYear": 0,
  "paymentMonth": 0,
  "paymentAmount": 0
}
```

- Yukarıdaki yapıya uyarak verilen yıl ve ay için verilen miktarda bir aidat faturası, bütün dairelere atanmaktadır.
- Burada dairelerin boş olmamasına ve verilen yıl ve aydaki aidatını ödememiş olmasına dikkat edilmektedir.

GetAllPayments

- Bu fonksiyonu sadece **admin** kullanabilir.
- Bir parametre vermeden, bütün kullanıcıların ödenmiş veya ödenmemiş bütün faturalar yazdırılabilmektedir.

GetPaymentsWithGivenFlatNumber

- Bu fonksiyonu **admin** veya **kullanici** rolündeki kişiler kullanabilir.

```
{
  "flatNo": 0,
  "isPaid": false,
  "paymentType": "aidat"
}
```

- Yukarıdaki yapı ile, numarası verilen dairenin verilen türdeki (**paymentType**), ödenmiş veya ödenmemiş (**isPaid**) faturaları görüntülenebilmektedir.

PayBillWithGivenFlatNumber

- Bu fonksiyonu **admin** veya **kullanici** rolündeki kişiler kullanabilir.

```
{
  "isCreditCard": true,
  "paymentYear": 2024,
  "paymentMonth": 2,
  "paymentType": "elektrik",
  "flatNo": 3
}
```

- Yukarıdaki şema ile ödeme bilgileri girilerek verilen numaralı dairenin **faturası ödenebilir**.

- Bu fonksiyonda yapılan kontroller:
    - Eğer bir kullanıcı bir önceki sene aidatlarını düzenli **(vaktinde ve eksiksiz)** ödedi ise ve şu anki ödemesinin tipi **aidat** ise %10 daha az ödeme yapar ve veritabanına ödeme %10 daha az ve ödenmiş olarak kaydedilir. **(BONUS Gerçekleştirildi)**
    - Eğer kullanıcı şu an gerçekleştirdiği ödemeyi zamanında gerçekleştirmiyor ise %10 daha fazla ödeme yapar. Ardından veritabanına %10 daha fazla ve ödenmiş olarak kaydedilir. **(BONUS Gerçekleştirildi)**

GetTotalPaymentAmountWithGivenFlatNumber

- Bu fonksiyonu **admin** veya **kullanici** rolündeki kişiler kullanabilir.
- Kullanıcı ödenmemiş faturalarının **toplam miktarını** aşağıdaki yapıya uyguna olarak bilgilerini verip görebilir.

```
{
  "flatNo": 3,
  "paymentYear": 2024,
  "paymentMonth": 2,
  "monthlyOrYearly": "Monthly",
  "paymentType": "elektrik"
}
```

  Not: Monthly veya Yearly yazarken baş harfi büyük yazılmalıdır.

GetUsersPayingRegularly

- Bu fonksiyonu sadece **admin** kullanabilir.

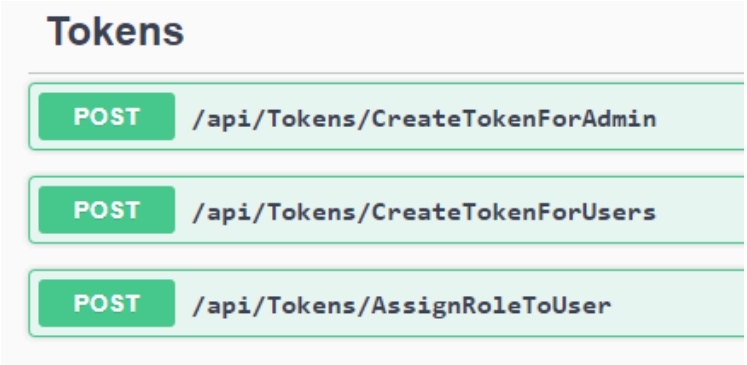
```
{
  "paymentType": "aidat",
  "paymentPeriod": 6
}
```

- Yukarıdaki yapıya uyarak örnekte gösterildiği üzere son 6 ayda (paymentPeriod) faturalarını **eksiksiz** ve **zamanında düzenli** olarak ödeyen kullanıcılar görüntülenebilir. **(BONUS Gerçekleştirildi)**

GetUsersHavingUnpaidAidatPaymentForCurrentMonth

- Bu fonksiyonu sadece **admin** kullanabilir.
- Herhangi bir parametre vermeden bu ay **aidat ödemesini** yapmayan kullanıcılar görüntülenebilir.

Token Fonksiyonları



- **CreateTokenForAdmin:** Bu fonksiyon anonim olarak kullanılabilir. Kullanıcı adı ve şifre girilerek token alma işlemi gerçekleştirilir. Şifreye sahip olan tek kullanıcı sistem açıldığında oluşturulan **admin** kullanıcısı olduğundan, bu fonksiyon ile token alabilecek tek kullanıcı admindir.

- **CreateTokenForUsers:** Bu fonksiyon anonim olarak kullanılabilir. TC Numarası ve Telefon Numarası girilerek token alma işlemi gerçekleştirilir. Kullanıcılar sistemde işlem yapabilmek için buradan token oluşturmalıdır.

- **AssignRoleToUser:** Bu fonksiyonu sadece **admin** kullanabilir. Oluşturulan bir kişiye, kişinin ID'si ve atanmak istenen rol verilerek o kişiye rol ataması yapılır.
  Eğer atanmak istenen rol sistemde bulunmuyorsa oluşturulur. Fakat **kullanici** veya **admin** hariç roller sistemdeki hiçbir fonksiyonu kullanamamaktadır

Zehra Göl
11.02.2024