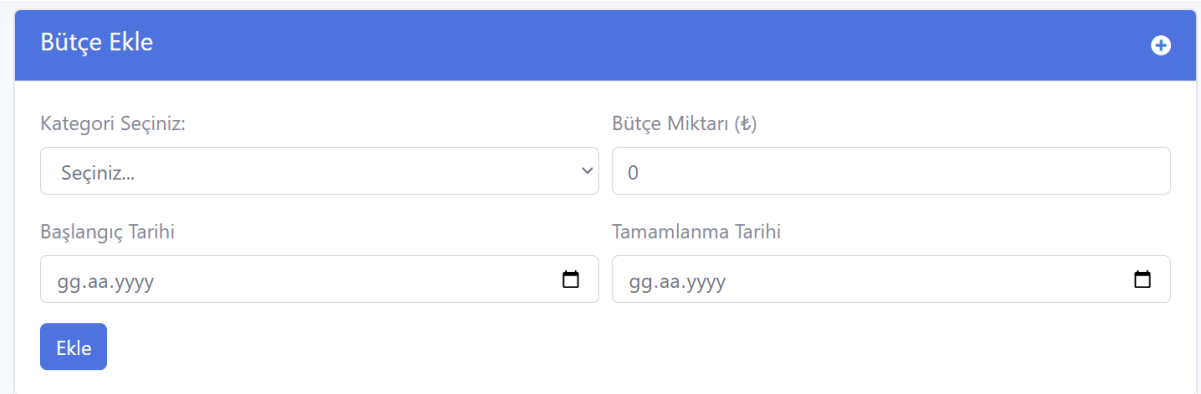
ANGULARDA CRUD İŞLEMLERİ

Bütçe için CRUD:




```
6 references
export class BudgetService {
  //  apiUrl: 'http://localhost:5177/api'
  private apiUrl = environment.apiUrl + '/Budget';
  0 references
  constructor(private http: HttpClient) { }

  getPosts(): Observable<any> {
    return this.http.get<any>(`${this.apiUrl}`);
  }
  getPostsBudgetByUser(userId: number): Observable<any> {
    return this.http.get<any>(`${this.apiUrl}/BudgetByUser/${userId}`);
  }

  createPost(post: any): Observable<any> {
    return this.http.post<any>(`${this.apiUrl}`, post);
  }

  updatePost(id: number, post: any): Observable<any> {
    return this.http.put<any>(`${this.apiUrl}/update/${id}`, post);
  }
  deletePost(id: number): Observable<any> {
    return this.http.delete<any>(`${this.apiUrl}/${id}`);
  }
  getPostById(id: number): Observable<any> {
    return this.http.get<any>(`${this.apiUrl}/budget/${id}`); // API URL'sini güncelleyin
  }

}
```

Bütçe ile ilgili API'nin temel URL'sini tutar.

HTTP istekleri yapmak için kullanılır

Observable: RxJS kütüphanesinden gelen bir yapı. Asenkron veri akışlarını temsil eder.

Tüm bütçe verilerini almak için bir GET isteği yapar.

BudgetService, bir Angular uygulamasında, bütçe ile ilgili CRUD (Create, Read, Update, Delete) işlemlerini gerçekleştirmek için gereken tüm temel işlevselliği sunar. Asenkron veri akışlarını yönetmek için RxJS Observable yapısını kullanarak, uygulamanın performansını ve kullanıcı deneyimini artırır.

Bu Angular servisi, bir bütçe yönetim uygulamasının arka uç API'si ile etkileşimde bulunmak için oluşturulmuştur.

```
export class BudgetComponent implements OnInit {

  categoryItems: { id: number, categoryName: string, categoryType: string }[] = [];
  items: { categoryID: number, amount: number, startDate: string, endDate: string, budgetId: number,
   categoryName: string }[] = [];
  newItem = { categoryID: 0, amount: 0, startDate: '', endDate: '', budgetId: 0, categoryName: '' };
  selectedItem: any=0; // Güncellenen öğeyi tutacak değişken
  isUpdateFormVisible: boolean = false; // Güncelleme formunun görünürlüğü

  0 references
  constructor(
    private renderer: Renderer2,
    private budgetService: BudgetService,
    private router: Router,
    private categoryApiService: CategoryApiService
  )
  {
    this.loadCategoryItems();
  }

  ngOnInit() {
    this.loadScriptsSequentially();

    this.loadCss('assets/vendor/datatables/dataTables.bootstrap4.min.css');

    this.loadItems(); // Sayfa yüklendiğinde bütçe öğelerini al
  }
```

- categoryItems: Kategorileri tutar (örneğin, harcama kategorileri).
- items: Kullanıcının bütçe öğelerini tutar.
- newItem: Yeni bir bütçe eklemek için kullanılan modeldir.
- selectedItem: Güncellenmekte olan öğeyi tutar.
- isUpdateFormVisible: Güncelleme formunun görünür olup olmadığını belirler.

- constructor: Servisleri (dependency injection) enjekte eder.
  - Renderer2: DOM ile etkileşim için.
  - BudgetService: Bütçe işlemleriyle ilgili API çağrıları için.
  - Router: Yönlendirme işlemleri için.
  - CategoryApiService: Kategori verilerini almak için.

ngOnInit Metodu

- Bileşen yüklendiğinde ilk olarak çalışır.
- loadScriptsSequentially() ve loadCss() ile gerekli stil ve script dosyaları yüklenir.
- loadItems() ile kullanıcıya ait bütçe öğeleri alınır.

```
private loadCategoryItems(): void {
  this.categoryApiService.getPosts().subscribe({
    next: (categories) => {
      this.categoryItems = categories.data.items; // API'den gelen veriyi items'a ata
      console.log("Categories loaded: ", this.items);

    },
    error: (err) => {
      console.error("Category loading failed: ", err);
    }
  });
}
loadItems() {
  this.budgetService.getPostsBudgetByUser(1).subscribe(response => {
    console.log('API Yanıtı:', response); // Yanıtı kontrol edin

    // Yanıtın içindeki data nesnesini kontrol edin
    if (response && response.data && Array.isArray(response.data.items)) {
      this.items = response.data.items; // items dizisine atama yapın
    } else {
      console.error('Beklenen dizi değil:', response);
      this.items = []; // Hata durumunda boş dizi atayın
    }
  }, error => {
    console.error('Hata:', error);
    this.items = []; // Hata durumunda boş dizi atayın
  });
}
```

loadCategoryItems()

- Amaç: API'den kategori verilerini alıp categoryItems dizisine atar.
- subscribe: Asenkron çağrının sonucu işlenir.
    - next: Başarılı veri alımında çalışır.
    - error: Hata durumunda çalışır.

loadItems()

- Amaç: Kullanıcıya ait bütçe öğelerini API'den alır.
- Kontrol: Verinin data.items formatında olup olmadığını kontrol eder.

```
budgetCreate() {
  this.budgetService.createPost(this.newItem).subscribe(response => {
    console.log('Başarıyla eklendi:', response);
    this.items.push({ ...this.newItem }); // Yeni öğeyi diziye ekle
    alert("Başarıyla Eklendi");
    this.newItem = { categoryID: 0, amount: 0, startDate: '', endDate: '', budgetId: 0,
     categoryName :''}; // Formu sıfırla
  }, error => {
    console.error('Ekleme hatası:', error);
  });
}

guncelle(item: any) {
  this.selectedItem = { ...item }; // Seçilen öğeyi kopyala
  this.isUpdateFormVisible = true; // Güncelleme formunu göster
}

budgetUpdate(sid: number, post: any) {
  this.router.navigate(['budgetUpdate', sid], { state: { post: post } });

}
```

- guncelle(item: any)

Amaç: Parametre olarak gelen `item` nesnesinin bir kopyasını oluşturur ve bunu `this.selectedItem` değişkenine atar.

- Neden kopya alınır? Doğrudan referansı kullanmak, öğe üzerinde yapılan değişikliklerin anında listeye yansımasına neden olabilir. Bu istenmeyen bir durumdur çünkü güncellemeyi iptal ederseniz, orijinal öğe değişmiş olabilir. Bu yüzden bir kopya alınır.

`budgetCreate`: Yeni öğeyi API'ye gönderir ve başarılı olursa `items` dizisine ekler.

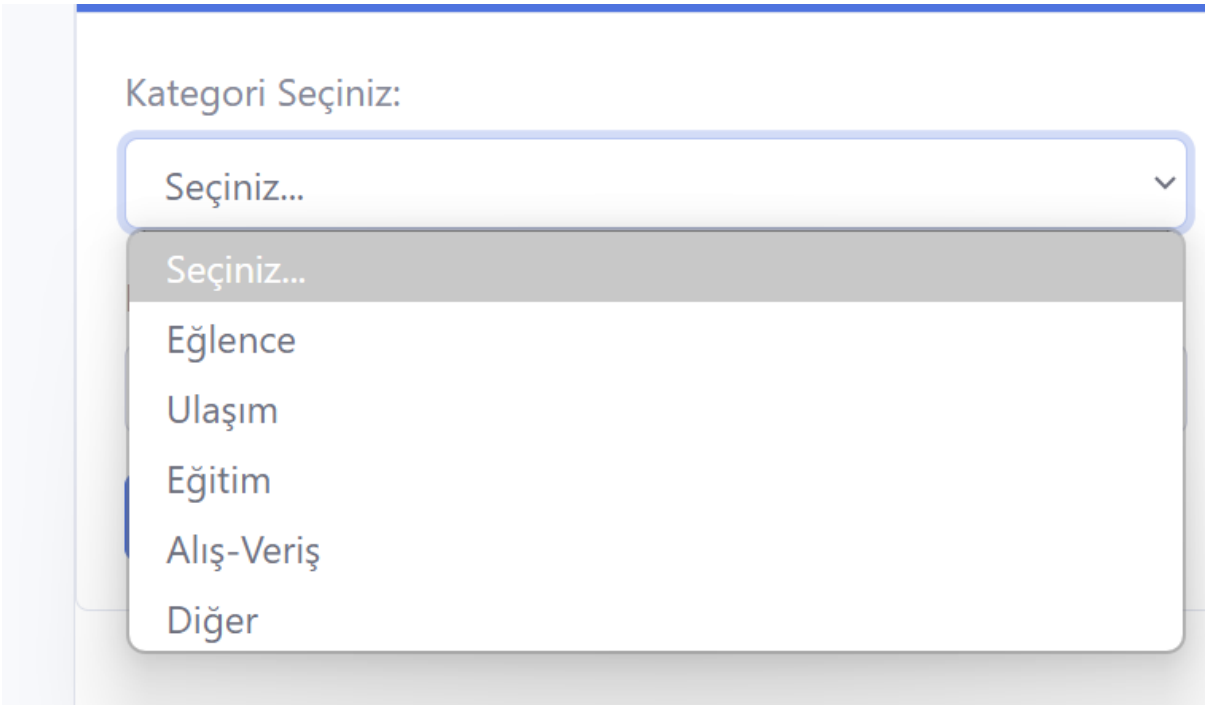
```
onUpdate(form: NgForm) {
  if (form.valid && this.selectedItem) {
    this.budgetService.updatePost(this.selectedItem.categoryID, this.selectedItem).subscribe(response => {
      console.log('Başarıyla güncellendi:', response);
      const index = this.items.findIndex(item => item.categoryID === this.selectedItem.categoryID);
      if (index !== -1) {
        this.items[index] = this.selectedItem; // Dizi içindeki öğeyi güncelle
      }
      this.isUpdateFormVisible = false; // Güncelleme formunu gizle
    }, error => {
      console.error('Güncelleme hatası:', error);
    });
  } else {
    console.log('Form geçersiz.');
  }
}

deleteHedef(itemId: number) {
  const confirmDelete = confirm('Bu öğeyi silmek istediğinize emin misiniz?');
  if (confirmDelete) {
    this.budgetService.deletePost(itemId).subscribe(response => {
      console.log('Başarıyla silindi:', response);
      this.items = this.items.filter(item => item.categoryID !== itemId); // Öğeyi diziden çıkar
    }, error => {
      console.error('Silme hatası:', error);
    });
  }
}
```

- onSubmit: Form geçerliyse budgetCreate çağrılır.

deleteHedef(itemId: number)
Amaç: Belirtilen öğeyi API üzerinden silmek ve `items` dizisinden kaldırmak.

- Onay: Kullanıcıdan silme işlemi için onay alır.

```
<div class="form-row">
  <div class="form-group col-md-6">
    <label for="categorySelect">Kategori Seçiniz:</label>
    <select [(ngModel)]="newItem.categoryID" id="categorySelect" name="categoryID"
    class="form-control" required>
      <option [value]="0" disabled>Seçiniz...</option>
      <option *ngFor="let item of categoryItems" [value]="item.id">
        {{ item.categoryName }}
      </option>
    </select>
  </div>
```


Kategori Seçiniz:
Seçiniz...
Seçiniz...
Eğlence
Ulaşım
Eğitim
Alış-Veriş
Diğer