GREEN CASE - FULL STACK DEVELOPER CASE STUDY DOKÜMANTASYONU

🎯 PROJE HAKKINDA

Proje Adı: Green Case - Müşteri Yönetim Sistemi

Framework: Laravel 12 + PHP 8.3

Frontend: Bootstrap 5 + jQuery + Vite

Veritabanı: MySQL (MSSQL uyumlu yapı)

Özel Kütüphaneler: DataTables, Laravel Sanctum, Yajra DataTables

📋 CASE STUDY GEREKSİNİMLERİ

Temel Gereksinimler:

- Laravel ve Blade kullanılarak geliştirilmeli
- Müşterilerin pagination ile sunucu taraflı (server-side) listelenmesi
- DataTables (datatables.net) kullanılarak tablolama
- 10,000 sahte müşteri verisi (Faker + Seeder)
- Users ve Customers tabloları
- Login sonrası müşteri listesi sayfasına yönlendirme
- Müşteri Ekleme, Güncelleme, Silme işlemleri

Tablo Kolonları:

- Müşteri Kodu
- Müşteri Adı
- Adres
- Telefon
- Email
- Oluşturan Kullanıcı
- Oluşturma Tarihi
- Güncelleyen Kullanıcı
- Güncelleme Tarihi
- İşlem butonları
- Pagination:
- Seçenekler: 10, 20, 30, 50, 100, Tümü
- Varsayılan: 100 kayıt

🚀 GELİŞTİRME AŞAMALARI (SIRA SIRA)

ADIM 1: VERITABANI YAPISI

1.1 Migration Dosyaları

- database/migrations/2014_10_12_000000_create_users_table.php
- database/migrations/2025_10_03_190648_create_customers_table.php
- database/migrations/2025_10_04_141902_add_deleted_by_to_customers_table.php
- database/migrations/2025_10_04_144317_add_delete_reason_to_customers_table.php
- database/migrations/2025_10_05_103550_add_deleted_at_to_users_table.php

1.2 Customers Tablosu Yapısı

Schema::create('customers', function (Blueprint $t) {

$t->id();

$t->string('code', 50)->unique(); // Müşteri kodu

$t->string('name')->index(); // Müşteri adı

$t->string('address')->nullable(); // Adres

$t->string('phone', 32)->nullable(); // Telefon

$t->string('email')->nullable()->index(); // Email

$t->foreignId('created_by')->nullable()->constrained('users'); // Oluşturan

$t->foreignId('updated_by')->nullable()->constrained('users'); // Güncelleyen

$t->foreignId('deleted_by')->nullable()->constrained('users'); // Silen

$t->string('delete_reason')->nullable(); // Silme nedeni

$t->timestamps(); // created_at, updated_at

$t->softDeletes(); // deleted_at (Soft Delete)

});

1.3 Users Tablosu Ek Özellikler

- role (enum: admin, manager, staff)

- deleted_at (Soft Delete)

ADIM 2: MODEL İLİŞKİLERİ

2.1 Customer Model (app/Models/Customer.php)

```
class Customer extends Model
{
use SoftDeletes;
protected $fillable = [
'code', 'name', 'address', 'phone', 'email',
'created_by', 'updated_by', 'deleted_by', 'delete_reason'
];
// İlişkiler
public function creator() {
return $this->belongsTo(User::class, 'created_by');
}
public function updater() {
return $this->belongsTo(User::class, 'updated_by');
}
public function deleter() {
return $this->belongsTo(User::class, 'deleted_by');
}
}
```

2.2 User Model (app/Models/User.php)

```
class User extends Authenticatable
{
use HasApiTokens, HasFactory, Notifiable, SoftDeletes;
protected $fillable = ['name', 'email', 'password', 'role'];
protected $hidden = ['password', 'remember_token'];
protected $casts = [
```

'email_verified_at' => 'datetime',

'password' => 'hashed',

];

}

ADIM 3: SEEDER OLUŞTURMA (10,000 VERİ)

DB::table('users')->insert([

['name' => 'Admin User', 'email' => 'admin@greenholding.com', 'password' =>
Hash::make('password'), 'role' => 'admin'],

['name' => 'Manager User', 'email' => 'manager@greenholding.com', 'password' =>
Hash::make('password'), 'role' => 'manager'],

['name' => 'Staff User', 'email' => 'staff@greenholding.com', 'password' =>
Hash::make('password'), 'role' => 'staff'],

]);

3.2 CustomerSeeder (database/seeders/CustomerSeeder.php)

// 10,000 kayıt - Batch Processing ile Optimize Edilmiş

$faker = \Faker\Factory::create('tr_TR');

$total = 10000;

$batchSize = 100;

for ($b = 0; $b < $total / $batchSize; $b++) {

$rows = [];

for ($i = 0; $i < $batchSize; $i++) {

$index = $b * $batchSize + $i + 1;

$rows[] = [

'code' => 'CMP' . str_pad($index, 6, '0', STR_PAD_LEFT),

'name' => $faker->company(),

'address' => $faker->address(),

'phone' => $faker->phoneNumber(),

'email' => $faker->unique()->companyEmail(),

'created_by' => $faker->randomElement($userIds),

'updated_by' => $faker->boolean(30) ? null : $faker->randomElement($userIds),

'created_at' => now(),

'updated_at' => now(),

];

}

DB::table('customers')->insert($rows);

}

Optimizasyon:

- Batch insertion (100'lük gruplar)
- Memory efficient
- Progress tracking

ADIM 4: AUTHENTICATION SİSTEMİ

4.1 AuthController (app/Http/Controllers/AuthController.php)

- Login sayfası
- Login işlemi
- Logout işlemi
- Session yönetimi

4.2 Middleware Yapılandırması

Route::middleware('auth')->group(function () {

// Tüm protected routes

});

ADIM 5: CUSTOMER CONTROLLER VE CRUD İŞLEMLERİ

5.1 CustomerController (app/Http/Controllers/CustomerController.php)

Metodlar:
index() - Müşteri listesi sayfası
return view('customers.index');

data() - DataTables için JSON API
$query = Customer::with(['creator', 'updater']);

return DataTables::of($query)
    ->addColumn('created_at', fn($c) => $c->created_at->format('d.m.Y H:i'))

```
    ->addColumn('updated_at', fn($c) => $c->updated_at->format('d.m.Y H:i'))
    ->addColumn('updater_name', fn($c) => $c->updater ? $c->updater->name : '-')
    ->addColumn('actions', function ($customer) {
        // Admin ve Manager silme yapabilir
        // Staff sadece görüntüleyebilir
    })
    ->make(true);

store() - Müşteri ekleme
$customer = Customer::create([
    'code' => uniqid('CUST-'),
    'name' => $request->name,
    'email' => $request->email,
    'phone' => $request->phone,
    'address' => $request->address,
    'created_by' => Auth::id(),
    'updated_by' => Auth::id(),
]);

update() - Müşteri güncelleme
$customer->update([
    'name' => $request->name,
    'email' => $request->email,
    'phone' => $request->phone,
    'address' => $request->address,
    'updated_by' => Auth::id(),  // Kim güncelledi tracking
]);




destroy() - Soft Delete
// Admin ve Manager silme yapabilir
if (!in_array(Auth::user()->role, ['admin', 'manager'])) {
    abort(403, 'Silme yetkiniz yok');
}

$customer->deleted_by = Auth::id();
$customer->delete_reason = $request->reason ?? 'Belirtilmedi';
$customer->save();
$customer->delete();  // Soft Delete
```

ADIM 6: DATATABLE ENTEGRASYONU

6.1 Frontend DataTables Yapılandırması (resources/js/pages/customers.js)

```
const table = $("#customers-table").DataTable({
    // Responsive ayarları
    responsive: {
        details: { type: 'column', target: 'tr' },
        breakpoints: [
            { name: 'desktop', width: Infinity },
            { name: 'tablet-l', width: 1024 },
            { name: 'tablet-p', width: 768 },
            { name: 'mobile-l', width: 480 },
            { name: 'mobile-p', width: 320 }
        ]
    },

    // Server-side processing
    processing: true,
    serverSide: true,
    ajax: "/customers/data",

    // Pagination ayarları
    pageLength: 100,
    lengthMenu: [
        [10, 20, 30, 50, 100, -1],
        [10, 20, 30, 50, 100, "Tümü"]
    ],

    // Kolonlar
    columns: [
        { data: "code", name: "code", responsivePriority: 1 },
        { data: "name", name: "name", responsivePriority: 2 },
        { data: "address", name: "address", responsivePriority: 7 },
        { data: "phone", name: "phone", responsivePriority: 6 },
        { data: "email", name: "email", responsivePriority: 5 },
        { data: "creator.name", name: "creator.name", responsivePriority: 8 },
        { data: "created_at", name: "created_at", responsivePriority: 9 },
        { data: "updater_name", name: "updater_name", responsivePriority: 10 },
        { data: "updated_at", name: "updated_at", responsivePriority: 4 },
        { data: "actions", name: "actions", responsivePriority: 3, orderable: false }
    ],

    // Türkçe dil desteği
    language: {
        lengthMenu: "Sayfada _MENU_ kayıt göster",
```

```
        info: "Toplam _TOTAL_ kayıttan _START_ - _END_ arası gösteriliyor",
        search: "Ara:",
        paginate: { previous: "Önceki", next: "Sonraki" }
    }
});
```

6.2 DataTable Reinitialize Koruması

```
// Her sayfa yüklendiğinde önceki instance'ı temizle
if ($.fn.DataTable.isDataTable($table)) {
    $table.DataTable().clear().destroy();
    $table.empty();
}
```

ADIM 7: RESPONSIVE TASARIM

7.1 CSS Yapılandırması (resources/scss/_datatables.scss)

```
// Masaüstü (1200px+)
@media (min-width: 1200px) {
    table.dataTable {
        font-size: 0.85rem;
        th, td {
            padding: 0.5rem 0.4rem;
            white-space: nowrap;  // Satır kırılmasını engelle
        }
    }
}

// Mobil (<1200px)
@media (max-width: 1199px) {
    table.dataTable tbody td {
        white-space: normal;  // Satır kırılabilir
    }
}

// Scrollbar styling
.table-wrapper {
    overflow-x: auto;
    -webkit-overflow-scrolling: touch;
}
```

7.2 Responsive Priority Sistemi
Priority 1-3: Her zaman görünür (Kod, Ad, İşlem)
Priority 4-6: Tablet'te görünür

Priority 7-10: Sadece masaüstünde

ADIM 8: ROLE-BASED ACCESS CONTROL (RBAC)

8.1 Roller ve Yetkiler

Rol	Müşteri Görme	Müşteri CRUD	Müşteri Silme	Silinen Kayıtları Görme	Kullanıcı Yönetimi
Admin	✅	✅	✅	✅	✅ Tam Yetki
Manager	✅	✅	✅	❌	✅Görüntüleme
Staff	✅	❌	❌	❌	❌

8.2 Controller Yetkilendirmeleri

CustomerController - destroy():

```
// Admin ve Manager silme yapabilir
if (!in_array(Auth::user()->role, ['admin', 'manager'])) {
    abort(403, 'Silme yetkiniz yok');
}
```

TrashController - _construct():

```
// Sadece admin silinenleri görebilir
$this->middleware(function ($request, $next) {
    if (Auth::check() && Auth::user()->role !== 'admin') {
        abort(403, 'Bu sayfaya erişim yetkiniz yok. Sadece admin erişebilir.');
    }
    return $next($request);
});
```

UsersController - destroy():

```
// Admin ve Manager silme yapabilir
if (!Auth::check() || !in_array(Auth::user()->role, ['admin', 'manager'])) {
    abort(403, 'Silme yetkiniz yok');
}

// Kendini silemez
if (Auth::id() == $id) {
    return response()->json(['message' => 'Kendi kullanıcınızı silemezsiniz!'], 403);
}
```

ADIM 9: DASHBOARD SİSTEMİ

9.1 DashboardController (app/Http/Controllers/DashboardController.php)

```
public function index()
{
    $user = Auth::user();
    $activeCustomers = Customer::count();

    // Sadece admin silinenleri görebilir
    $deletedCustomers = ($user->role === 'admin')
        ? Customer::onlyTrashed()->count()
        : null;

    // Admin ve Manager kullanıcıları görebilir
    $users = in_array($user->role, ['admin', 'manager'])
        ? User::count()
        : null;

    return view('dashboard', compact('activeCustomers', 'deletedCustomers', 'users'));
}
```

9.2 Dashboard View - Dinamik Col Yapısı (resources/views/dashboard.blade.php)

```
@php
    $role = Auth::user()->role;
    if ($role === 'admin') {
        $colClass = 'col-md-4';  // 3 kart
    } elseif ($role === 'manager') {
        $colClass = 'col-md-6';  // 2 kart
    } else {
        $colClass = 'col-md-8 offset-md-2';  // 1 kart (ortalanmış)
    }
@endphp

<!-- Admin: Aktif Müşteriler + Silinen Kayıtlar + Kullanıcılar -->
<!-- Manager: Aktif Müşteriler + Kullanıcılar -->
<!-- Staff: Sadece Aktif Müşteriler →
```

ADIM 10: TRASH (SOFT DELETE) SİSTEMİ

10.1 TrashController (app/Http/Controllers/TrashController.php)

Özellikler:

- Silinen müşterileri listeleme
- Silinen kullanıcıları listeleme
- Geri yükleme (restore)
- Kalıcı silme (force delete)

dataCustomers() Metodu:

```
$query = Customer::onlyTrashed()->with(['creator', 'updater', 'deleter']);

return DataTables::of($query)
    ->addColumn('created_at', fn($c) => $c->created_at->format('d.m.Y H:i'))
    ->addColumn('updated_at', fn($c) => $c->updated_at->format('d.m.Y H:i'))
    ->addColumn('deleted_at', fn($c) => $c->deleted_at->format('d.m.Y H:i'))
    ->addColumn('creator_name', fn($c) => $c->creator?->name ?? '-')
    ->addColumn('updater_name', fn($c) => $c->updater?->name ?? '-')
    ->addColumn('deleter_name', fn($c) => $c->deleter?->name ?? '-')
    ->make(true);
```

10.2 Silinen Müşteriler Sayfası

13 Kolon:

- Müşteri Kodu
- Müşteri Adı
- Adres
- Telefon
- Email
- Oluşturan Kullanıcı
- Oluşturma Tarihi
- Güncelleyen Kullanıcı
- Güncelleme Tarihi
- Silen Kullanıcı (ekstra)
- Silinme Tarihi
- Silme Nedeni (ekstra)
- İşlem (Geri Yükle + Kalıcı Sil)

ADIM 11: KULLANICI YÖNETİMİ

11.1 UsersController (app/Http/Controllers/UsersController.php)

Özellikler:

- Kullanıcı listeleme (Admin + Manager)
- Kullanıcı ekleme (Sadece Admin)
- Kullanıcı güncelleme (Sadece Admin)
- Kullanıcı silme (Admin + Manager)
- Kendi kullanıcısını silememe koruması

data() Metodu - Kendini Silme Koruması:

```
->addColumn('actions', function ($row) {
    $buttons = '<button class="btn btn-primary btn-sm editUserBtn">Düzenle</button>';

    // Kendi kullanıcısı için Sil butonunu gösterme
    if ($row->id != Auth::id()) {
        $buttons .= ' <button class="btn btn-danger btn-sm deleteUserBtn">Sil</button>';
    }

    return $buttons;
})
```

ADIM 12: PROFİL YÖNETİMİ

12.1 ProfileController (app/Http/Controllers/ProfileController.php)

Özellikler:

- **update() - Profil bilgileri güncelleme**
  - Ad soyad güncelleme
  - Email güncelleme
  - Validation (email unique kontrolü)
- **updatePassword() - Şifre değiştirme**
  - **Güvenlik Katmanları:**
    - Mevcut şifre zorunlu
    - Mevcut şifre doğrulama (Hash::check)
    - Yeni şifre != Eski şifre kontrolü
    - Yeni şifre confirmation (tekrar)
    - Minimum 6 karakter

12.2 Profil Sayfası (resources/views/profile/edit.blade.php)

İki Kartlı Yapı:

- Kart 1 - Kişisel Bilgiler:
  - Ad Soyad (düzenlenebilir)
  - E-posta (düzenlenebilir)
  - Rol (sadece görüntüleme)
  - Kayıt Tarihi (sadece görüntüleme)
- Kart 2 - Şifre Değiştir:
  - Mevcut Şifre (zorunlu)
  - Yeni Şifre (min 6 karakter)
  - Yeni Şifre Tekrar (eşleşmeli)

ADIM 13: FRONTEND AJAX İŞLEMLERİ

13.1 AJAX DELETE İşlemleri - Method Spoofing
Sorun: jQuery AJAX DELETE metodu bazı sunucularda çalışmaz

Çözüm: Laravel method spoofing

```
// ❌ YANLIŞ
$.ajax({
    url: '/customers/1',
    type: 'DELETE'
})

// ✅ DOĞRU
$.ajax({
    url: '/customers/1',
    type: 'POST',
    data: {
        _token: csrf_token,
        _method: 'DELETE'
    }
})
```

Uygulanan Dosyalar:

- resources/js/pages/customers.js
- resources/js/pages/users.js
- resources/js/pages/trashedCustomers.js
- resources/js/pages/trashedUsers.js

ADIM 14: BLADE VIEW YAPISI

14.1 Layout Sistemi (resources/views/layouts/app.blade.php)

Navbar Menü Yapısı (Role-based):
// Herkes görebilir

- Dashboard
- Müşteriler

// Sadece Admin görebilir

- Silinen Kayıtlar (dropdown)
- Silinen Müşteriler
- Silinen Kullanıcılar

// Admin + Manager görebilir

- Kullanıcılar

// Herkes için
- Profil (dropdown)
- Profil Bilgilerim
- Çıkış Yap

14.2 Müşteri Sayfaları

resources/views/customers/
- index.blade.php (Liste)
- create.blade.php (Ekleme)
- edit.blade.php (Düzenleme)
- partials/form.blade.php (Ortak form)

14.3 Kullanıcı Sayfaları

resources/views/users/
- index.blade.php (Liste)
- create.blade.php (Ekleme)
- edit.blade.php (Düzenleme)

14.4 Trash Sayfaları

resources/views/trash/
- trashed_customers.blade.php (Silinen müşteriler)
- trashed_users.blade.php (Silinen kullanıcılar)

14.5 Profil Sayfası

resources/views/profile/
- edit.blade.php (Profil düzenleme + Şifre değiştirme)

ADIM 15: JAVASCRIPT MODÜLLERİ

15.1 Dosya Yapısı

resources/js/
- app.js (Ana entry point)
- bootstrap.js (Bootstrap konfigürasyonu)
- datatables.js (Reusable DataTable helper)

pages/

- customers.js (Müşteri listesi)
- trashedCustomers.js (Silinen müşteriler)
- users.js (Kullanıcı listesi)
- trashedUsers.js (Silinen kullanıcılar)
- users_create.js (Kullanıcı ekleme)
- users_edit.js (Kullanıcı düzenleme)
- customerForm.js (Müşteri form işlemleri)
- profile.js (Profil sayfası)

utils/
- toast.js (Bildirim sistemi)

15.2 Toast Notification Sistemi (resources/js/utils/toast.js)

```
export function showToast(type, message) {
    // Bootstrap Toast ile bildirimler
    // Tipler: success, error, delete, add, edit
}
```

ADIM 16: SASS/SCSS YAPISI

resources/scss/
- app.scss (Ana dosya)
- variables.scss (Renk paleti, spacing)
- navbar.scss (Navbar stilleri)
- datatables.scss (DataTables custom styles)
- responsive.scss (Responsive breakpoints)
- custom.scss (Özel stiller)

pages/
- dashboard.scss (Dashboard kartları)
- trashed.scss (Trash sayfaları)

ADIM 17: ROUTING YAPISI

```
// Guest Routes
Route::get('/login')->name('login');
Route::post('/login')->name('login.post');

// Protected Routes (Auth Middleware)
Route::middleware('auth')->group(function () {

    // Müşteri İşlemleri
    Route::resource('customers', CustomerController::class);
```

```
    // Silinen Kayıtlar (Admin Only)
    Route::get('/trash/customers', ...);
    Route::get('/trash/users', ...);
    Route::post('/trash/{type}/{id}/restore', ...);
    Route::delete('/trash/{type}/{id}/force-delete', ...);

    // Kullanıcı İşlemleri
    Route::get('/users', ...);
    Route::middleware(['can:manage-users'])->group(function () {
        Route::resource('users', UsersController::class)->except(['index']);
    });

    // Profil
    Route::get('/profile', ...);
    Route::post('/profile/update', ...);
    Route::post('/profile/update-password', ...);

    // Logout
    Route::post('/logout', ...);
});

// Default Route
Route::get('/', [DashboardController::class, 'index']);
```

ADIM 18: POLİCY SİSTEMİ

18.1 UserPolicy (app/policies/UserPolicy.php)

```
public function manageUsers(User $user)
{
    return $user->role === 'admin';
}

public function update(User $user, User $model)
{
    return $user->role === 'admin';
}

public function delete(User $user, User $model)
{
    return $user->role === 'admin';
}
```

18.2 AuthServiceProvider (app/Providers/AuthServiceProvider.php)

```
protected $policies = [
    \App\Models\User::class => \App\Policies\UserPolicy::class,
];

Gate::define('manage-users', [\App\Policies\UserPolicy::class, 'manageUsers']);
```

ADIM 19: VITE YAPILANDIRMASI

19.1 package.json

```
{
    "dependencies": {
        "bootstrap": "^5.3.8",
        "bootstrap-icons": "^1.13.1",
        "datatables.net": "^2.3.4",
        "datatables.net-bs5": "^2.3.4",
        "datatables.net-responsive": "^3.0.7",
        "datatables.net-responsive-bs5": "^3.0.7",
        "jquery": "^3.7.1",
        "sass": "^1.93.2"
    }
}
```

19.2 vite.config.js

```
export default defineConfig({
    plugins: [
        laravel({
            input: ['resources/scss/app.scss', 'resources/js/app.js']
        })
    ]
});
```

ADIM 20: ÖZEL ÖZELLİKLER (BONUS)

20.1 Soft Delete Sistemi

- Customers ve Users için soft delete
- deleted_by tracking (Kim sildi?)
- delete_reason (Neden silindi?)
- Geri yükleme özelliği
- Kalıcı silme seçeneği

20.2 Audit Trail

- created_by (Kim oluşturdu?)
- updated_by (Kim güncelledi?)
- deleted_by (Kim sildi?)
- Tarih takibi (created_at, updated_at, deleted_at)

20.3 Modal Sistemleri

- Silme onay modalı (Radio button ile neden seçimi)
- Bootstrap Modal entegrasyonu
- AJAX işlemleri

20.4 Toast Notification

- Success bildirimler
- Error mesajları
- Delete işlemi bildirimleri
- Position: top-right

📦 KULLANILAN PAKETLER

Backend (composer.json):

```
{
    "php": "^8.3",
    "laravel/framework": "^12.0",
    "laravel/sanctum": "^4.1",
    "yajra/laravel-datatables-oracle": "^12.5"
}
```

Frontend (package.json):

```
{
    "bootstrap": "^5.3.8",
    "jquery": "^3.7.1",
    "datatables.net-bs5": "^2.3.4",
    "datatables.net-responsive-bs5": "^3.0.7",
    "sass": "^1.93.2",
    "vite": "^5.0.0"
}
```

🎨 UI/UX ÖZELLİKLERİ

Tasarım Kararları:

- Bootstrap 5 - Modern ve responsive
- Green Tema - Navbar ve primary color yeşil (#198754)
- Card-based UI - Dashboard kartları
- Dark table headers - Görsel hiyerarşi
- Hover effects - Kullanıcı geri bildirimi
- Icons - Bootstrap Icons kullanımı
- Shadow effects - Depth algısı

Responsive Breakpoints:

- Desktop: 1200px+
- Tablet: 768px - 1199px
- Mobile: < 768px

🔧 TEKNIK DETAYLAR

Best Practices:

- MVC Architecture - Temiz kod ayrımı
- Repository Pattern (Eloquent ORM)
- Service Layer (Controller'da business logic minimum)
- Form Requests - Validation
- Policy Pattern - Authorization
- Soft Delete Pattern - Veri kaybı önleme
- Eager Loading - N+1 query problemi önleme
- CSRF Protection - Güvenlik
- Method Spoofing - RESTful API
- Modüler JavaScript - ES6 modules

Performance Optimizations:

- Server-side DataTables - 10,000 kayıt için zorunlu
- Batch seeding - 100'lük gruplar halinde insert
- Database indexing - name, email, code kolonları
- Lazy loading - Vite code splitting
- Eager loading - with(['creator', 'updater'])

🗂️ PROJE DOSYA YAPISI

```
green-case/
├── app/
│   ├── Http/
│   │   └── Controllers/
│   │       ├── AuthController.php
│   │       ├── CustomerController.php
│   │       ├── UsersController.php
│   │       ├── TrashController.php
│   │       ├── DashboardController.php
│   │       └── ProfileController.php
│   ├── Models/
│   │   ├── Customer.php
│   │   └── User.php
│   └── Policies/
│       └── UserPolicy.php
│
├── database/
│   ├── migrations/ (5 migration dosyası)
│   └── seeders/
│       ├── UserSeeder.php
│       └── CustomerSeeder.php (10,000 kayıt)
│
├── resources/
│   ├── views/
│   │   ├── layouts/
│   │   │   └── app.blade.php
│   │   ├── customers/ (index, create, edit, partials)
│   │   ├── users/ (index, create, edit)
│   │   ├── trash/ (trashed_customers, trashed_users)
│   │   ├── profile/
│   │   │   └── edit.blade.php
│   │   └── dashboard.blade.php
│   │
│   ├── js/
│   │   ├── pages/ (8 sayfa JS dosyası)
│   │   └── utils/
│   │       └── toast.js
│   │
│   └── scss/
│       ├── app.scss
│       ├── _datatables.scss
│       ├── _navbar.scss
│       ├── _responsive.scss
│       └── pages/
│
├── routes/
│   ├── web.php
│   └── api.php
│
└── public/
```

🔐 GÜVENLİK ÖZELLİKLERİ

1. Authentication
- laravel Sanctum
- Session-based auth
- Password hashing (bcrypt)
- Remember token

2. Authorization
- Role-based access control
- Gate ve Policy kullanımı
- Middleware korumas ı
- Frontend buton gizleme
- Backend API koruması

3. CSRF Protection
- Her AJAX isteğinde CSRF token
- Meta tag ile token yönetimi

4. Input Validation
- Server-side validation
- Client-side validation (HTML5)
- Custom validation messages (Türkçe)

5. SQL Injection Koruması
- Eloquent ORM kullanımı
- Prepared statements
- Parameter binding

📊 VERITABANI İSTATİSTİKLERİ

Tablolar:
- users: 3 kayıt (admin, manager, staff)
- customers: 10,000 kayıt (faker ile)

İlişkiler:
- customers → users (created_by)
- customers → users (updated_by)
- customers → users (deleted_by)

İndeksler:
- customers.code (UNIQUE)
- customers.name (INDEX)
- customers.email (INDEX)

💻 KURULUM VE ÇALIŞTIRMA

1. Gereksinimler:

- PHP 8.3+
- Composer
- Node.js 18+
- MySQL 8.0+
- Laragon / XAMPP / Laravel Valet

2. Kurulum Adımları:

```
# Bağımlılıkları yükle
composer install
npm install

# .env dosyasını yapılandır
cp .env.example .env
php artisan key:generate

# Veritabanını oluştur ve migrate et
php artisan migrate

# Seed data (10,000 müşteri + 3 kullanıcı)
php artisan db:seed

# Vite build
npm run dev

# Serveri başlat
php artisan serve
```

3. Test Kullanıcıları:
Admin:

- Email: admin@greenholding.com
- Password: password
- Yetki: Tam yetki

Manager:

- Email: manager@greenholding.com
- Password: password
- Yetki: CRUD + Soft Delete (silinenleri göremez)

Staff:

- Email: staff@greenholding.com
- Password: password
- Yetki: Sadece görüntüleme