

**KAN BAĞIŞI MERKEZİ VERİTABANI
VE
WEB PORTALI
2012**

**Emin EKER
Dilara KOCA**

version 1.0

İçindekiler

1. Genel Bilgiler	3
Proje Tanımı	3
Kapsam	4
Kullanım Senaryosu Modeline İlişkin İnceleme	4
Tanımlar, Kısa Adlar ve Kısaltmalar	4
İş Paylaşımı	4
Geliştirme Ortamı	5
Neden Ruby on Rails	5
Profesyonel Yapı ve Görünüm	5
2. Kullanım Kılavuzu	6
Anasayfa	6
Donör Kayıt Ekranı	7
Kurum Kayıt Ekranı	8
Yardım Ekranı	9
Giriş Ekranı	10
Misafir Donör Sorgu Ekranı	11
Kurumlar Ekranı	11
Donör Paneli	13
Kurumlar Paneli	15
Yönetim Paneli	19
3. Kurulum	23
Çalıştırma Ortamı (Gereksinimler)	23
Veritabanı Kurulumu	23
Veritabanı Oluşturulması	24
Rails Kurulumu	24
Servisin Kurulması	25
Veritabanı-Servis Konfigürasyonu	25
Servisin Çalıştırılması	27
Servis Bağımlılıkları	27
4. Teknik Kılavuz	28
Veritabanı Tasarımı	28
Veritabanı Tabloları	31
Tablo İlişkileri	37
Yazılım Tasarımı	38

Bölüm 1. Genel Bilgiler

Proje Tanımı

Kan bankalarının ve hastanelerin yakın çevrede olan ve kısa zamanda kan bağıışı yapabilecek kişileri bulabilecekleri çevrimiçi, merkezileştirilmiş bir web portalı oluşturulacak. Ayrıca, hızla değerlendirmek üzere kan bağıışı yapan kişilerin sağlık raporlarının kaydını tutacaktır.

Kan bağıışı merkezi veritabanı ve web portalı kan bağıışçılarının gönüllülükleri ile doğru orantılı olarak kan bağıışlarının kayıt altına alınması ve ihtiyaçlar doğrultusunda (acil durumlarda) sisteme daha önceden kayıt yaptırmış ve onay olmuş kurumlar tarafından email veya telefon aracılığı ile kendilerine anında ulaşılmasını sağlayacaktır. Projenin temel özellikleri aşağıda verilmiştir.

Sistemde dört çeşit kullanıcı olacaktır. Bunlar; donör, misafir, kurum ve yönetici kullanıcılarıdır. Bütün kullanıcılar sisteme kullanıcı adı ve şifresi ile giriş yapabilecektir. Kullanıcı tiplerinin özellikleri şöyledir:

Donör: Kan grubu, ad, soyad, cinsiyet, doğum günü, iletişim bilgileri ve parola bilgilerini doğru olarak doldurarak kayıt olunur. Kayıt isteği yönetici tarafından onaylandıktan sonra sisteme giriş yapılabilir. Kaydı onaylanan kullanıcı, giriş yaptıktan sonra bilgilerini düzenleyebilir, parolasını değiştirebilir.

Misafir: Sisteme giriş yapmasına gerek yoktur. Arama yapmak istediği il ve aradıkları kan grubunu girerek Donör Sorgulama özelliğini kullanabilir. Sisteme kayıtlı olan kurum ve kuruluşları görebilir.

Kurum: Adı, tipi, iletişim bilgileri ve parola bilgilerini doğru olarak doldurarak kayıt olunur. Kullanıcının kayıt işleminden sonra sisteme giriş yapabilmesi için yönetici tarafından gerçek bir kurum veya kuruluş olduğu görülmeli ve başvurusu onaylanmalıdır. Kaydı onaylanan kullanıcı, Kurum Girişi yaptıktan sonra kendi kurum bilgilerini düzenleyebilir, parolasını değiştirebilir. Arama yapmak istediği il ve aradıkları kan grubunu girerek Donör Sorgulama özelliğini kullanabilir.

Yönetici: Yönetici Girişi yaptıktan sonra yöneticileri, kurum rollerini, kurumları, donörleri ve onlara ilişkin tüm ayrıntıları, ayrıca donörlerin özel isteklerini, kayıtlı kan verme işlemlerini, sorgulama sayısını görebilir.

Kapsam

Öncelikler belirli bir bölgeyi kapsayacak bu proje daha sonra tüm ülkeye yayılıp merkezi bir sistem olarak gerekli kurum ve kuruluşlara hizmet sağlayacaktır.

Kullanım Senaryosu Modeline İlişkin İnceleme

Donör, sisteme tamamen kendi isteğiyle kayıt olabilir ve sistem üzerinde bir parolaya sahip olur. Donör aynı zamanda sistem kullanıcı adı olacak email adresi ve parolasıyla sisteme giriş yapar. Sistemde içerisinde kayıt olurken verdiği bilgileri düzenleyebilir, parolasını değiştirebilir. Donör ve kurumlar açısından önemli olan bir diğer faktör kullanıcının sağlık bilgileri ve kan verme tarihleridir. Donör yine bu bilgileri de güncelleyebilmektedir.

Kurum , öncelikle kurum adları ve kurum email adresleri ile sisteme başvuruda bulunurlar. Yapılan bu başvuru anında sistem yöneticilerinin önüne düşer ve sistem yöneticileri bu başvurunun gerçek kurum veya kuruluşlar tarafından yapıp yapılmadığını araştırarak kurum başvurusunu onaylar veya reddeder. Başvurusu onaylanmış kurum veya kuruluşlar email adresleri ve parolaları ile sisteme giriş yapabilir yapabilirler. Sistem dahilinde kurum bilgilerini düzenleyebilir, parolalarını değiştirebilirler. Kurumların sisteme girişlerinden sonra sisteme dahil olmuş kan bağışçıları kan grubu, kan grubu il, kan grubu, il, ilçe, üzerinden sorgulayabilirler ve aldıkları sonuçlardan aradıkları özelliklere sahip donörlerin tüm bilgilerine erişebilirler. Acil kan ihtiyacı doğrultusunda anında çevredeki bağışçılara ulaşabilirler ve kan ihtiyaçlarını karşılayabilirler. Bu gibi kurumlar tarafından sistem dahilindeki donörler tarafından alınan kanların kaydı yine bu kurumlar tarafından sistem üzerinde yapılabilmektedir.

Tanımlar, Kısa Adlar ve Kısaltmalar

Donör: Gönüllü kağ bağışçısı

Kurum: Sürekli veya acil kan ihtiyacı olan devlet kurumu veya özel kuruluşlar (hastaneler, klinikler vb.)

Admin: Sistem yöneticisi

İş Paylaşımı

Emin EKER:

Veri tabanı tasarımı
Sistemin Gerçeklenmesi

Dilara KOCA:

Veri tabanı tasarımı
Proje Testleri
Belgelendirme (Raporlandırma)

Geliştirme Ortamı

Proje MySQL veritabanı ve Ruby programlama dili kullanılarak Ruby on Rails web çatısı üzerinde Linux işletim sistemi Debian dağıtımı üzerinde geliştirilmiştir. Projede ayrıca Java Script kodları da yer kullanılmıştır. Web sunucusu olarak ise Nginx kullanılmıştır. Aynı zamanda production bir Rails sistemi çalıştırabilmek için Unicorn Rails sunucusu kullanılmıştır. Proje geliştiricileri tarafından ayrı ortamlarda hazırlanmıştır.

Tüm geliştiriciler:

```
ruby >= 1.9.2
rails >= 3.1.0
Nginx 1.1.17
MySQL 5.1.61
unicorn v4.2.0
imagemagick görüntü işleme paketi
```

sürümlerini kullanmışlardır.

Proje geliştirilme esnasında Chromium browser 19.01.1077 üzerinde çalışılmış olup diğer tüm tarayıcılarda test edilmiştir.

Neden Ruby on Rails

Ruby on Rails (RoR) geliştirici verimliliğini artırmak için optimize edilmiştir. Bu uygulama geliştiricinin normal geliştirme zaman diliminin yarısında zengin web uygulamaları oluşturmaya yardım etmek için birçok gelişmiş programlama konsepti kullanır. Her şeyden önce RoR geliştiricilere bir dizi avantaj sunmaktadır;

- Hızlı geliştirme modelini destekler,
- Perl ve Python destekli tutarlı ve basit nesne yönelimli dil,
- CGI veya Fast CGI ile herhangi bir web sunucusunda çalışır,
- MVC mimari uygulaması programlama senkronize,
- Gelişmiş ilkeler kullanır: Kurulum Sözleşmesi CoC ve DRY,
- Kısa zaman dilimleri içerisinde uygulama geliştirme için idealdir,
- MySQL, PostgreSQL, Oracle gibi veritabanları için entegre destek,
- Hata ayıklama, özel URL'ler ve geniş Ajax kütüphanesi.

Profesyonel Yapı ve Görünüm

Ruby on Rails ortamında hazırlanan bu serviste görsel tasarım açısından Twitter'ın da kullandığı bir açık kaynak servis olan bootstrap kullanıcı arayüzü kullanılmıştır. Bootstrap kullanıcı arayüzü hem bilgisayar ortamında hem tabletlerde hem de telefonlarda sorunsuz bir görüntünün elde edilmesini amaçlar. Bu gibi açık kaynak ürünlerin rails ortamına daha hızlı adapte edilebilmesi için ruby gem geliştirilmiştir. Bu gem aracılığıyla bu kullanıcı arayüzünü servisimize adapte ettik ve kullandık.

Daha ayrıntılı bilgiyi buradan edinebilirsiniz:

<http://twitter.github.com/bootstrap/>

Bölüm 2. Kullanım Kılavuzu

Kan bağıışı merkezi veritabanı ve web portalı; kan bağıışçıların ve kan bağıışına gereksinin duyan kurumların tek bir merkezden kontrol edilebilmesini sağılayan bölümleri aşğıdaki gibidir:

1. **Ana Sayfa:** Sistem kullanıcılarının şifre ve kullanıcı adı ile giriş yapabileceğı, sisteme üye olmayanların üye olabileceğı, aynı zamanda misafir kullanıcıların örnek donörler arayabileceğı temel ekrandır.
2. **Donör Kayıt Ekranı:** Kan bağıışçıların gönüllü kayıt olabildikleri ekrandır.
3. **Kurum Kayıt Ekranı:** Kurumların kayıt olabildikleri ekrandır.
4. **Yardım Ekranı:** Donör, kurum ve yöneticilerin sistem kullanımı için yardım aldıkları ekranlardır.
5. **Giriş Ekranı:** Kayıtlı kullanıcıların sisteme giriş ekranlarıdır.
6. **Misafir Sorgu Ekranı:** Kayıtlı olmayan kurum veya kuruluşların örnek sorgulama ekranıdır.
7. **Kurumlar Ekranı:** Kayıtlı kurum veya kuruluşların gösterildiğı ekrandır.
8. **Donör Paneli:** Donörlerin kişisel bilgilerini düzenleyip sistemle etkileşime geçtikleri ekranlardır.
9. **Kurumlar Paneli:** Kurumların bilgilerini düzenleyip donör sorgulayabildikleri ekranlardır.
10. **Yönetici Paneli:** Sistemi yöneten, düzenleyen ve tüm sistem bilgilerine erişilebilen ekranlardır.

Ana Sayfa:

Donör Merkezi


Anasayfa

Hakkında

İletişim

Giriş Yap


Kan Bağıışı Hayat Kurtarır



Haydi Kan Bağıışına

Sürekli kan bağıışında bulunmak ve zor durumlarda size ulaşılmasını istiyorsanız sisteme kayıt olabilirsiniz.


Donör olmak istiyorum



Donör Sorgula

İhtiyacınız olan kan grubunu buradan sorgulayabilir sistem yöneticileri aracılığıyla kan bağıışçılarına ulaşabilirsiniz.

Donör Sorgula



Kayıtlı Kuruluşlar

İhtiyaç anında sistemden anında faydalanabilen kurum ve kuruluşları görebilir siz de onlardan biri olabilirsiniz.

Kayıtlı Kuruluşlar

Kan Bağıışçısı Sayısı: 29, Kurum Sayısı: 2

Sistemi çalıştıran kullanıcı, işlemlerine ilk olarak bu sayfadan başlar.

Sayfadaki “Donör olmak istiyorum” butonu kullanıcıyı **Donör Kayıt Ekranı**’na, “Donör Sorgula” butonu **Misafir Sorgu Ekranı**’na, “Kayıtlı Kuruluşlar” butonu ise **Kurumlar Ekranı**’na yönlendirmektedir. Sağ üstteki “Giriş Yap” butonu ise seçim yapılmasının ardından **Giriş Ekranı**’na yönlendirir.

Donör Kayıt Ekranı:

Kan Bağışı için Kayıt Olun

Kan Grubu (*)

Tc Kimlik Numarası (*)

Ad (*)

Soyad (*)

Cinsiyet (*)

☒ Erkek

☐ Bayan

Doğum Tarihi (*)

Örneğin: 1990-04-05

Fotoğraf

No file chosen

İletişim Bilgileri

E-mail (*)

Telefon Numarası

Örneğin: 0506-693-4002

İl (*)

İlçe (*)

Ana Sayfa'dan "Donör olmak istiyorum" butonu ile bu ekrana ulaşılmaktadır.

Bu ekranda kan bağışçıları, kişisel bilgilerini girerek sisteme kayıt olabilirler.

Formda, kan grubu, ad, soyad, il, ilçe, e-mail adresi alanlarının doldurulması zorunludur. Ayrıca sisteme kayıt olmak isteyen donörün gerçek bir kişi olup olmadığının belirlenmesi amacıyla tc kimlik numarası, cinsiyet, doğum günü bilgilerinin de zorunlu tutulması gerektiği düşünülmüştür ve sistem bu şekilde gerçekleştirilmiştir. Diğer alanların doldurulması zorunlu değildir.

Kayıt yönetici tarafından onaylanana kadar kullanıcı sisteme giriş yapamaz. Kayıt onaylandıktan sonra verdiği bilgileri düzenleyebilir.

Kurum Kayıt Ekranı:

Daha Hızlı Donör Bulmak için Kayıt Olun

Kurum Adı (*)

Kurum Tipi (*)

Fotoğraf

Choose File

No file chosen

İletişim Bilgileri

E-mail (*)

Telefon Numarası

Örneğin: 0506-693-4002

İl (*)

İlçe (*)

Adres

Kurumlar Ekranı'ndan "Yeni Kurum Kaydı" butonu ile bu ekrana ulaşılmaktadır.

Kurum ve Kuruluşlar bu ekranda kurum bilgilerini girerek sisteme kayıt olabilirler.

Formda, kayıt olmak isteyen kurum veya kuruluşun gerçek olup olmadığının belirlenmesi amacıyla kuruluş adı, kurum tipi, e-mail adresi, il, ilçe alanlarının doldurulması zorunludur. Diğer alanların doldurulması zorunlu değildir.

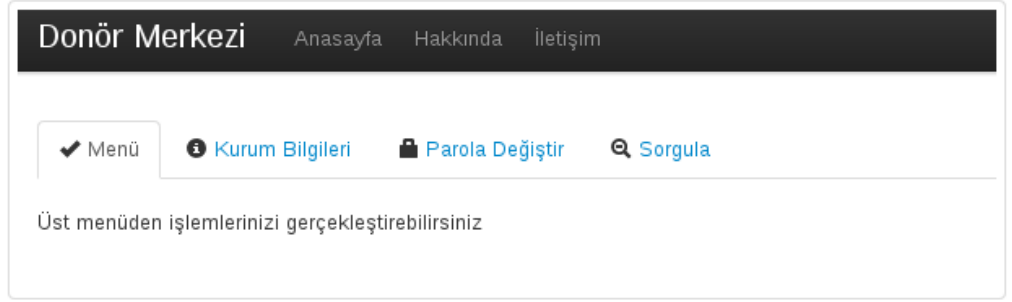
Kayıt yönetici tarafından onaylanana kadar kullanıcı sisteme giriş yapamaz. Kayıt onaylandıktan sonra verdiği bilgileri düzenleyebilir.

Yardım Ekranı:

Kullanıcı girişi yaptıktan sonra görülen “Yardım” butonundan ulaşılmaktadır.

Kullanılan bu paneline raporda “Kullanım Kılavuzu” başlığı altında anlatılan kısmını ekrana getirir.
Örnek ekran görüntüsü:

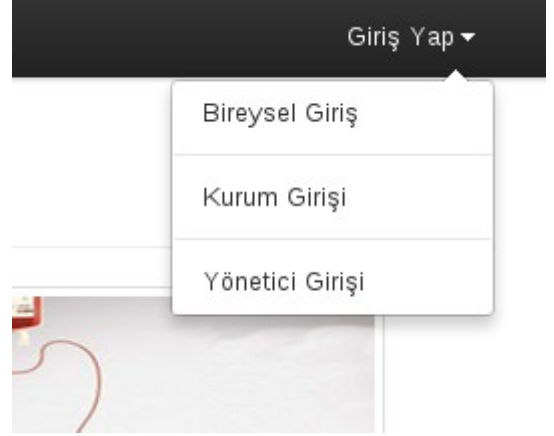
Kurum Yardım Sayfası



Kurum sisteme giriş yaptığında böyle bir ekran ile karşılaşır.
Burada kurum bilgilerini değiştirebilir, parolasını güncelleyebilir ve donör sorgulayabilir.

Giriş Ekranı:

“Giriş Yap” butonu üç seçenek sunmaktadır:
“Bireysel Giriş”, “Kan Bağışçısı Giriş Ekranı”na;
“Kurum Girişi”, “Kurum Giriş Ekranı”na;
“Yönetici Girişi” ise “Admin Giriş Ekranı”na yönlendirir.



“Kan Bağışçısı Giriş Ekranı”: Sisteme donör olarak kayıtlı olan kullanıcı için giriş ekranıdır.

“Kurum Giriş Ekranı”: Sisteme kurum olarak kayıtlı olan kullanıcı için giriş ekranıdır.

“Admin Giriş Ekranı”: Sisteme yönetici olarak kayıtlı olan kullanıcı için giriş ekranıdır. Örnek ekran görüntüsü:

Donör Merkezi [Anasayfa](#) [Hakkında](#) [İletişim](#)

Kan Bağışçısı Girişi

E-mail

Parola

Giriş Yap

Donör Sorgu Ekranı:

Anasayfa'ndan "Donör Sorgula" butonu ile bu ekrana ulaşılmaktadır.

Misafirler kullanıcılar da kan grubu il ve/veya ilçe seçerek sistemde sorgu yapabilirler. Sorgu sonucunda bulunan uygun donörler hakkında yalnızca cinsiyet, kan grubu, il ve ilçe bilgilerini görebilirler. Başka hiçbir ayrıntıya erişemezler.

Örnek bir "Misafir Donör Sorgu Ekranı" görüntüsü:

Donör Sorgulama

Kan Grubu

il

ilçe

Sorgula

Kan bağışçısı olarak kayıtlı ve giriş yapmış kullanıcı da il ve/veya ilçe seçerek sistemde sorgu yapabilirler. Ancak bu sorgu sonucunda bulunan uygun donörler hakkında sadece varolma bilgilerini görebilirler.

Kurum olarak kayıtlı ve giriş yapmış kullanıcı da il ve/veya ilçe seçerek sistemde sorgu yapabilirler. Bu sorgu sonucunda bulunan uygun donörler hakkında tüm bilgilerini görebilirler.

Kurumlar Ekranı:

Kurum ve Kuruluşlar

[Yeni Kurum Kaydı](#)

Adı	il	ilçe	Kurum Tipi
YEŞİL YURT	ARTVİN	ARHAVİ	Klinik
Ondokuz Mayıs Üniversitesi Tıp Fakültesi	SAMSUN	ATAKUM	Hastane

Anasayfa'ndan "Kayıtlı Kuruluşlar" butonu ile bu ekrana ulaşılmaktadır.

Misafirler kullanıcılar bu sistemde yer alan kurum veya kuruluşlardan sadece örnek olması amacıyla birkaç tanesini görebilirler.

Donör Paneli

Kan Bağışçısı Girişi yapıldığında bu ekran ile karşılaşılacaktır.

✓ Menü

❗ Kişisel Bilgiler

🔒 Parola Değiştir

📄 Sağlık Bilgileri

📍 Acil Kan Bağışı İsteği

Üst menüden işlemlerinizi gerçekleştirebilirsiniz

Menüde:

"Kişisel Bilgiler": sisteme kayıt olunurken verilen bütün bilgiler düzenlenebilir. Örnek bir ekran görüntüsü:

✓ Menü

❗ Kişisel Bilgiler

🔒 Parola Değiştir

📄 Sağlık Bilgileri

📍 Acil Kan Bağışı İsteği

Kullanıcı Bilgileri Güncelleme Ekranı

Kan Grubu (*)

A Rh(-)

Tc Kimlik Numarası (*)

32977301484

Ad (*)

Emin

Soyad (*)

EKER

Cinsiyet (*)

☒ Erkek

☐ Bayan

Doğum Günü (*)

1990-05-04

Fotoğraf



Choose File

No file chosen

“Parola Deęiřtir”: parola deęiřtirilebilir. Örnek bir ekran görüntüsü:

[✓ Menü](#) [i Kiřisel Bilgiler](#) [Parola Deęiřtir](#) [Saęlık Bilgileri](#) [Acil Kan Baęıřı İsteęi](#)

Parola Bilgileri

Kullandığınız Parola (*)

Yeni Parola (*)

Yeni Parola Tekrar (*)

Parolamı Deęiřtir

İptal

“Saęlık Bilgileri”: Son kan baęıřı tarihi güncellenebilir. Örnek bir ekran görüntüsü:

[✓ Menü](#) [i Kiřisel Bilgiler](#) [Parola Deęiřtir](#) [Saęlık Bilgileri](#) [Acil Kan Baęıřı İsteęi](#)

Saęlık Bilgileriniz

Son Kan Verme Tarihiniz (*)

Bilgileri Kaydet

İptal

“Acil Kan Bağıışı İsteğı”: Acil kan bağıışı isteğı gönderilebilir. Donör herhangi bir kan bağıışı yardımı için başka bir kan bağıışçısı aramaları gerektiğinde bu formu doldurur. Form yöneticiye iletilecektir. Nedenin geçerliliğine göre sonuç alınacaktır. Örnek ekran görüntüsü:

[✓ Menü](#)[i Kişisel Bilgiler](#)[Parola Değıştir](#)[Sağılık Bilgileri](#)[Acil Kan Bağıışı İsteğı](#)

Acil Kan Bağıışı İstek Formu

Kan Grubu (*)

Bağıış İsteğinin Gerekçeleri ile Birlikte Açıkça Belirtiniz

İsteğı Gönder

İptal

Kurumlar Paneli

Kurum Girişi yaptığında bu ekran ile karşılaşılacaktır.

✓ Menü

i Kurum Bilgileri

🔒 Parola Değiştir

🔍 Sorgula

Üst menüden işlemlerinizi gerçekleştirebilirsiniz

Menüde:

"Kurum Bilgileri": sisteme kayıt olunurken verilen bütün bilgiler düzenlenebilir. Örnek bir ekran görüntüsü:

✓ Menü

i Kurum Bilgileri

🔒 Parola Değiştir

🔍 Sorgula

Kurum Bilgileri Güncelleme Ekranı

Kurum Adı (*)

Ondokuz Mayıs Üniversitesi Tıp Fakültesi

Kurum Tipi (*)

Hastane

Fotoğraf

Choose File

No file chosen

İletişim Bilgileri

E-mail (*)

info@omu.edu.tr

Telefon Numarası

Örneğin: 0506-693-4002

İl (*)

SAMSUN

İlçe (*)

ATAKUM

Adres

Kurupelit

Bilgilerimi Güncelle

İptal

“Parola Değiştir”: parola değiştirilebilir. Örnek bir ekran görüntüsü:

[✓ Menü](#) [i Kurum Bilgileri](#) [Parola Değiştir](#) [Sorgula](#)

Parola Bilgileri

Kullandığınız Parola (*)

Yeni Parola (*)

Yeni Parola Tekrar (*)

Parolamı Değiştir

İptal

“Sorgula”: Kan grubu il ve/veya ilçe seçerek sistemde sorgu yapabilirler. Örnek bir ekran görüntüsü:

[✓ Menü](#) [i Kurum Bilgileri](#) [Parola Değiştir](#) [Sorgula](#)

Kan Grubu

İl

İlçe

Sorgula

Kurumlar Paneli'nde yapılan sorgulamada, diğer kullanıcıların yaptığı sorgulamalardan farklı olarak, uygun donörlerin adı, soyadı ve donörlerin sistemdeki puanları da görülmektedir. Örnek ekran görüntüsü:

[✓ Menü](#) [i Kurum Bilgileri](#) [Parola Değiştir](#) [Sorgula](#)

A Rh(+)

SAMSUN

İlçe

Sorgula

Sorguyu Pdf'e Dök

#	Ad	Soyad	Cinsiyet	Kan Grubu	İl	İlçe	Puan	Göster
6	elçin duygu	karaca	Bayan	A Rh(+)	SAMSUN	SAMSAT	0	i
25	ali	uslucan	Erkek	A Rh(+)	SAMSUN	ÇİFTLİKKÖY	0	i
30	Erinç	Bereket	bayan	A Rh(+)	SAMSUN	ATAKUM	0	i
34	gizem	herguner	Bayan	A Rh(+)	SAMSUN	ATAKUM	0	i

Yapılan sorgu sonuçları yalnızca son 3 ay içinde kan vermemiş olan kişileri göstermektedir.

Ayrıca sistem sorgulamada, her üç aydan sonra aynı kan bağışçısını seçmek yerine, her defasında farklı bir kişiyi seçebilecek kadar akıllıdır.

“Sorguyu Pdf’e Dök” butonu, yapılan sorgu sonucu elde edilen uygun donörler listesini pdf formatında dışarı aktarır.

“Donör İncele”: Uygun donörler listesinin her satırında göster butonu bulunmaktadır.

Bu buton ile iki sekmeli yeni bir menü açılır.

“Donör Bilgileri”: Donörün fotoğraf, kan grubu, adı, soyadı, puan, e-mail, telefon numarası, cinsiyet, en son kan verme tarihi, doğum tarihi, il, ilçe, adres bilgileri ekrana getirilir. Örnek ekran görüntüsü:

[✓ Menü](#) [i Kurum Bilgileri](#) [🔒 Parola Değiştir](#) [🔍 Sorgula](#) [🔍 Donör İncele](#)

[Donör Bilgileri](#) [Kan Verme Kayıt İşlemi](#)

	Kan Bağışçısı Bilgileri
Fotoğraf	Fotoğraf Yüklememiş
Kan Grubu	A Rh(+)
Tc Kimlik Numarası	21353084400
Ad	gizem
Soyad	herguner
Puan	0
E-mail	gizemherguner@gmail.com
Telefon Numarası	
Cinsiyet	Bayan
En Son Kan Verme Tarihi	2011-12-30
Doğum Tarihi	1990-02-27
il	SAMSUN

“Kan Verme Kayıt İşlemi”: Kurumların donörlerden aldıkları kan bağışını raporlamaları için gerekli formu ekrana getirir. Örnek ekran görüntüsü:

[✓ Menü](#) [i Kurum Bilgileri](#) [🔒 Parola Değiştir](#) [🔍 Sorgula](#) [🔍 Donör İncele](#)

[Donör Bilgileri](#) **Kan Verme Kayıt İşlemi**

Kan Verme Kayıt İşlemi

Kurum (*)

Ondokuz Mayıs Üniversitesi Tıp Fakültesi

Dönör (*)

gizem herguner

Kan Verme Tarihi (*)

Açıklama Ekleyebilirsiniz

Kaydet

Her kan bağışına dayalı olarak, kan veren kişilerin akrabaları, arkadaşları ve diğer yakınları için kan gerekmesi durumunda kan bağışçıları tarafından kullanılacak bir puan verimelidir. Verilen puan, bu kan bağışçıları için üye kan bankaları tarafından kan bulunmasında belirli bir öncelik sağlamaktadır.

Yönetim Paneli

Yönetici Girişi yaptığında bu ekran ile karşılaşılacaktır. Kayıtlar ve sayılarını gösteren bir liste görülmektedir.

Yönetim		
Kayıtlar	Sayı	
Yöneticiler	2	Listele Ekle
Kurum Roller	3	Listele Ekle
Kurumlar	2	Listele Onay Bekleyen Yeni Bir Kurum Var
Donörler	29	Listele Onay Bekleyen Yeni Bir Donör Var
Donör Özel İstekleri	1	Listele
Kayıtlı Kan Verme	0	Listele
İstatistikler		
Tüm İstatistikler		Listele
Rollere Göre Kurum ve Kuruluş İstatistikleri		Listele
Kan Grubuna Göre Donör İstatistikleri		Listele
İllere Göre Donör İstatistikleri		Listele
Sorgulama Biçimleri		
Misafir Sorgulaması Sayısı		75
Kurum Sorgulaması Sayısı		14
Kurum Pdf Dökümü		2

“Yöneticiler”: Sisteme kayıtlı olan yöneticiler incelenebilir ve varlığı veya bilgileri düzenlenebilir.

“Ekle”: Sisteme yeni yönetici kaydı ekleme formunu ekrana gelir. Ad, soyad, email bilgileri doldurulması zorunlu alanlardır.

 [Listele](#)

 [Göster](#)

 [Yeni Ekle](#)

 [Düzenle](#)

Ad (*)

Soyad (*)

E-mail (*)

Telefon Numarası

Example: 0506-693-40-02

Fotoğraf

[Choose File](#)

No file chosen

Parola Bilgileri

Parola (*)

Parola Tekrar (*)

[Yönetici Ekle](#)







“Listele”: Yöneticilerin ad, soyad, e-mail bilgileri ekrana gelir.

 [Listele](#)

 [Göster](#)

 [Yeni Ekle](#)

 [Düzenle](#)

#	Adı	Soyadı	E-mail	
1	Emin	Eker	emineker@bil.omu.edu.tr	  
2	Dilara	Koca	dilara@bil.omu.edu.tr	  

[Yeni Yönetici Ekle](#)

Sağda bulunan butonlar ile:

“Göster”: Seçilen yöneticinin ad, soyad, e-mail, telefon numarası bilgileri ekrana gelir. Bilgilerin üzerinde değişiklik yapılamaz.

 [Listele](#)

 [Göster](#)

 [Yeni Ekle](#)

 [Düzenle](#)

Ad (*)

Dilara

Soyad (*)

Koca

E-mail (*)

dilara@bil.omu.edu.tr

Telefon Numarası


[Düzenle](#)

“Düzenle”: Seçilen yöneticinin ad, soyad, e-mail, telefon numarası, fotoğraf bilgileri ekrana gelir. Bilgiler güncellenilebilmektedir.

 [Listele](#)

 [Göster](#)

 [Yeni Ekle](#)

 [Düzenle](#)

Adı (*)

Dilara

Soyadı (*)

Koca

E-mail (*)

dilara@bil.omu.edu.tr

Telefon Numarası

Example: 0506-693-40-02

Fotoğraf

[Choose File](#)

No file chosen

[Bilgileri Güncelle](#)

“Sil”: Yöneticinin kaydını silinir. Silme işlemi gerçekleşmeden önce uyarı mesaj gönderilir:

Donör Merkezi Anasayfa Hakkında İletişim Yönetim Yardım emineker@bil.omu.edu.tr

Yönetim / Yöneticiler

Listele Göster Yeni Ekle

Bu kaydı gerçekten silmek istiyor musunuz?

İptal Tamam

#	Adı	Soyadı	E-mail	
1	Emin	Eker	emineker@bil.omu.edu.tr	1 Sil
2	Dilara	Koca	dilara@bil.omu.edu.tr	1 Sil

Yeni Yönetici Ekle

Bölüm 3. Kurulum

Projenin kurulumunu anlatırken herşeyden önce bir web sunucusu gereksinimlerinin sağlandığını düşünüyoruz. Dolayısıyla Nginx gibi sunucu paketlerinin sunucuda kurulu olmasını ve ayarlarının yapılmış olmasını bekliyoruz.

Çalıştırma Ortamı (Gereksinimler):

Çalıştırma ortamı olarak Unix tabanlı bir işletim sistemi dağıtımı olabilir. Servis için gerekli temel paketler şunlardır:

```
ruby >= 1.9.2
rails >= 3.1.0
Nginx 1.1.17
MySQL 5.1.61
unicorn v4.2.0
imagemagick görüntü işleme paketi
```

Veritabanı Kurulumu

Sunucuya MySql paketini kuruyoruz:

```
$ sudo apt-get install mysql-server-5.1 mysql-client-5.1
```

Kurulumda bizden mysql-server için root şifresi istiyor bunları girip kurulumu bitiriyoruz.

mysql processinin çalışıp çalışmadığını kontrol ediyoruz:

```
$ sudo ps aux | grep mysqld
```

```
mysql 3799 0.5 3.6 145392 18484 ? Ssl 10:42 0:00 /usr/sbin/mysqld
hummer 3891 0.0 0.1 3208 808 pts/0 S+ 10:43 0:00 grep --color=auto mysqld
```

Veritabanı Oluşturulması

Öncelikle projenin veritabanı bağlantısının gerçekleştirilebilmesi için sunucu üzerindeki mysql'de yeni bir veritabanı ve yeni bir veritabanı kullanıcısı açılır.

```
$ mysql -u root -p
```

komutu ve parolası ile veritabanına bağlanılır.

mysql root parolamızı girdikten sonra parolamızı giriyoruz ve mysql promptunu görüyoruz.

```
mysql>
```

Promptu gördükten sonra artık mysql üzerinde işlem yapabiliriz. Yapmamız gereken yeni bir veritabanı ile yeni bir kullanıcı oluşturmak;

```
// kan adında bir veritabanı oluşturalım:
```

```
mysql> create database kan;
```

```
// mysql üzerinde sadece 'kan' veritabanına erişim hakları olan
```

```
// 'kan' adında yeni bir kullanıcı oluşturalım
```

```
mysql> create user 'kan'@'% 'identified by 'parola';
```

```
// yetkisini sadece 'kan' veritabanı üzerine yönlendirelim
```

```
mysql> grant all on kan.* to 'kan'@'% ';
```

```
// yeni kullanıcı işlemini tamamlayalım:
```

```
mysql> flush privileges;
```

Rails Kurulumu

```
$ sudo apt-get install rails
```

Komutu ile rails paketini kuruyoruz. Bu paket rails'ın güncel bir sürümü olarak 3.2 sürümünü kuracaktır.

rails sunucusu olması için ise bir de unicorn paketini kuralım:

```
$ sudo apt-get install unicorn
```


Servisin Kurulması

Her bir uygulamanın bir git deposu olduğunu düşünürsek ve bu şekilde bir servis kurulumu yaparsak işler çok kolaylaşır.

Proje servisinin sunucuya indirelim:

```
$ cd /opt  
$ sudo git clone git://github.com/emineker/kan.git
```

Servis gereksinimleri için gerekli gem paketlerini otomatik kuralım:

```
$ cd /opt/kan  
$ bundle install
```

Bu komut bizden gerekli gemlerin kurulabilmesi için sistem parolamızı isteyecektir.

Veritabanı-Servis Konfigürasyonu

Yukarıda oluşturduğumuz veritabanı kullanıcı adı ve parolasını kuracağımız rails uygulamamıza aktarmamız gerekiyor. Bunun için (servisin kök dizininde olduğumuzu düşünürsek) config/database.yml dosyasına gerekli bilgileri ekliyoruz

```
$ vim config/database.yml
```

```
production:  
  adapter: mysql2  
  encoding: utf8  
  reconnect: false  
  database: kan  
  pool: 5  
  username: kan  
  password: parola  
  socket: /var/run/mysqld/mysqld.sock
```

Gördüğümüz gibi servisin veritabanı bağlantısı için daha önce veritabanı ayarlarında kullanıcıyı oluşturduğumuz veritabanı bilgilerini sisteme aktardık.

Şimdi servisimizin gerekli veritabanı işlemlerini yapalım. Eğer sistemde daha önce kullanılan bir 'kan' veritabanı var iste bunu silelim ve yenisini oluşturalım:

```
$ rake db:drop  
$ rake db:create
```

Şimdi de veritabanı tablolarımızı veritabanına yükleyelim:

```
$ rake db:migrate
```

Bu şekilde artık veritabanında gerekli tablolarımız mevcut hale geldi. Bu kısımdan sonra asıl önemli olan default verilerin yüklenmesi. Örneğin biz default olarak sistemin bir admin'inin olmasını istiyoruz. O halde db/seeds.rb dosyasına bakalım:

```
if Admin.count == 0
  Admin.create(:email => 'emineker@bil.omu.edu.tr', :password => '123456')
  Admin.create(:email => 'dilara@bil.omu.edu.tr', :password => '123456')
end
```

Dosyada gördüğümüz gibi bu admin'ler sistem kurulduğunda öntanımlı değerlerimizden birisi olacak. Daha sonra elbette silebiliriz. Ancak servisin çalıştırılabilmesi için bu bilgiler gerekli. Şimdi bu scripti çalıştıralım:

```
$ rake db:seeds
```

Bu şekilde sisteme öntanımlı değerleri yükledik. Bir de il ve ilçe listelerimiz var bunlar için ise hazır sql dosyalarımızı bulunmakta bunları da veritabanına yükleyelim:

```
$ mysql -u kan -p kan < src/city-districts-list.sql
```

Arkasından da parolamızı girdikten sonra src/city-districts-list.sql dosyasını veritabanına yüklemiş olduk.

Rails; servisleri içerisine hazır betikler yazıp kullanmamıza olanak sağlar. Servisin her güncellemesinden sonra veya yeni bir kurulumdan sonra tüm bunları yeniden yapmamak için bu işler için yazdığımız bir betik var. Servis kurulumları için hazırladığımız betiği çalıştıralım: bu bir Rakefile

```
$ rake x:db
```

Bu betik ile sistemin veritabanı ayarlarının tümünü yapmış oluyoruz.

Servisin Çalıştırılması

Daha önce rails servisinin sunucularda çalıştırılması için unicorn gibi bir rails sunucusuna ihtiyaç duyulduğunu söylemiştik. Şimdi bu ayarları yapalım:

Servis kök dizininde config/unicorn.rb dosyasına şu bilgileri girelim:

```
$ vim config/unicorn.rb
```

```
app_path = "/opt/kan"
```

```
listen 3000 # by default Unicorn listens on port 8080
listen 3001
worker_processes 2 # this should be >= nr_cpus
pid "#{app_path}/tmp/pids/unicorn.pid"
stderr_path "#{app_path}/log/unicorn.log"
stdout_path "#{app_path}/log/unicorn.log"
```

Daha sonra unicorn'a servisimizin nerede olduğunu bildirelim '/etc/unicorn' dizinine bir kan.conf soyası ekleyerek içine şunları ekleyelim:

```
$ vim /etc/unicorn/kan.conf
```

```
RAILS_ROOT=/opt/kan
RAILS_ENV=production
```

Servis Bağımlılıkları

```
$ sudo apt-get install imagemagick
```

Eğer sunucu uzun zamandır güncellenmemiş ise `libmagickwand-dev` paketine de ihtiyaç duyulabilir.

Bölüm 4. Teknik Kılavuz

Veritabanı Tasarımı

Veri tabanı tasarımı Rails'ın sunduğu ActiveRecord özelliğinin daha aktif kullanılabilmesi için yine Rails ortamında hazırlanmıştır.

Örneğin servisimizi hazırlarken kullanacağımız tabloları yavaş yavaş şekillendirmek için çeşitli işlemler yapılır. Öncelikle kullanacağımız tablo ile uyumlu bir isim seçilir ve bunun üzerinden gidilir. Bunu bir Örnekle açıklayalım:

```
$ rails generate model donor
```

Bu komutu verdikten sonra rails uygulaması otomatik olarak modelimizi, migrate dosyamızı ve test dosyalarını oluşturacaktır.

```
$ ls -al app/models/  
toplam 4  
-rw-r--r-- 1 dila dila 253 Mar 15 14:23 donor.rb
```

```
$ ls -al db/migrate/  
toplam 40  
-rw-r--r-- 1 dila dila 395 Mar 15 14:23 20120315184354_create_donors.rb
```

Gördüğümüz gibi kullanacağımız model modeller dizininde, migrate dosyamızda migrate dizininde oluşturuldu.

create_donors.rb dosyamızda şu kodları göreceğiz:

```
class CreateDonors < ActiveRecord::Migration  
  
  def change  
    create_table :donors do |t|  
  
      t.timestamps  
    end  
  end  
end
```

Biz ise bunu bu şekilde hazırlıyoruz:

```
class CreateDonors < ActiveRecord::Migration

  def change
    create_table :donors do |t|
      t.integer :tc
      t.integer :blood_group_id, :null => false

      t.string :first_name, :null => false
      t.string :last_name, :null => false

      t.string :email, :null => false
      t.string :password_digest, :null => false

      t.string :phone_number, :null => false

      t.string :image
      t.string :gender, :null => false
      t.date :birthday, :null => false
      t.date :blood_making_date

      t.integer :status, :default => 2
      t.integer :point
      t.integer :city_id, :null => false
      t.integer :district_id, :null => false
      t.string :address

      t.timestamps
    end
  end
end
```

Migrate dosyamızı oluşturduktan sonra bu dosyanın sql'e gönderme aşamasında ise şu komutu veriyoruz:

```
$ rake db:migrate
```

Bu komut tüm migrate dosyalarımızı MySql'e göndermekle görevlidir.

app/model/donor.rb dosyamızda şu kodları göreceğiz:

```
class Donor < ActiveRecord::Base
end
```

Biz ise bu modelimizi bu şekilde hazırlıyoruz:

```
class Donor < ActiveRecord::Base

  belongs_to :city
  belongs_to :district
  belongs_to :blood_group
  has_one :donor_request
  has_one :blood_making

  has_secure_password

  validates_presence_of :email, :first_name, :last_name
  validates_uniqueness_of :email

  def self.authenticate(email, password)
    find_by_email(email).try(:authenticate, password)
  end
end
```

Modelimizde ise bunlar oluyor. Bunları ilk olarak yapmak zorunda değiliz sistemi hazırlama şeklimize göre modelleri oluşturup söylebiliriz.

Dikkat: Biz modelimizi 'donor' diyerek oluşturduk. Bu model dosyamızda ise 'Donor' diye geçmektedir. Ruby'de nesneler hep büyük harfle başlar ve sistem bunu otomatik algılamaktadır. Migrate dosyamızda ise 'create donors' diye oluşmuş durumda. bizim Donor modelimiz veritabanında donors tablosunu yani kelime olarak çoğul halini kullanır ve bu işlemleri kendisi otomatik yapmaktadır.

Veritabanı tasarımını gerçekleştirdikten sonra model ve migrate dizinlerinin son hali

```
$ ls app/models
admin.rb
blood_group.rb
blood_making.rb
city.rb
district.rb
donor.rb
donor_request.rb
institute.rb
role.rb
search.rb

$ ls db/migrate
20120315184354_create_admins.rb
20120315185045_create_institutes.rb
20120315185345_create_donors.rb
20120315205327_create_roles.rb
20120317125452_create_cities.rb
20120317125812_create_blood_groups.rb
20120318192918_create_districts.rb
20120319164542_create_searches.rb
20120319183308_create_donor_requests.rb
20120321215002_create_blood_makings.rb
```

Bu şekilde örnek bir model oluşturduktan sonra şimdi veritabanı tablolarının açıklanmasına geçebiliriz.

Veritabanı Tabloları:

Biraz önce örnek olarak verdiğimiz donors tablosunda da gördüğümüzü gibi bir 'id' sütunu oluşturmadık. Daha doğrusu biz uğraşmadık rails bizim adımıza bunu yapıyor. Diğer bir husu ise 't.timestamps' sütunu. Biraz önce gördüğümüz gibi bu sütun model dosyamızın ilk halinde vardı yani bir öntanımlı değer. Bu timestamps sütunu dat formatında 'created_at' ve 'updated_at' sütunlarını oluşturmaktadır. İstersek bu sütunu silebiliriz de. Ancak veriler üzerinde işlem yapmak için bir tarih zaman bilgisini nasıl olsa tutacaktık. Bir veritabanı klasiği olduğundan Rails bunu öntanımlı bir değer olarak tutmaktadır.

1. Kan Bağışçıları Tablosu: donors

Field	Type	Null	Default	Key	Extra
id	int(8)	NO	PRI	NULL	auto_increment
tc	int(11)	NO	PRI	NULL	
blood_group_id	int(2)	NO		NULL	
first_name	varchar(32)	NO		NULL	
last_name	varchar(32)	NO		NULL	
email	varchar(64)	NO	PRI	NULL	
password_digest	varchar(128)	NO		NULL	
phone_number	varchar(16)	YES		NULL	
image	varchar(32)	YES		NULL	
gender	varchar(8)	NO		NULL	
birthday	date	NO		NULL	
blood_making_date	date	YES		NULL	
status	int(2)	NO		NULL	
point	int(4)	YES		NULL	
city_id	int(4)	NO		NULL	
district_id	int(4)	NO		NULL	
address	varchar(255)			2	
created_at	datetime	NO		NULL	
updated_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- tc: Kan bağışçısının kimlik numarası bilgisini tutar.
- blood_group_id: Kan bağışçısının kan gurubu bilgisini tutar.
- first_name: Kan bağışçısının adı bilgisini tutar.
- last_name: Kan bağışçısının soyadı bilgisini tutar.

- email: Kan bağışçısının email adresi bilgisini tutar.
- password_digest: Kullanıcı parolasının hashlenmiş 128 karakterlik bilgisini tutar.
- phone_number: Kan bağışçısının telefon numarası bilgisini tutar.
- image: Kullanıcı resim yolu bilgisini tutar.
- gender: Kan bağışçısının cinsiyet bilgisini tutar.
- birthday: Kan bağışçısının doğum günü bilgisini tutar.
- blood_making_date: Kan bağışçısının son kan verme tarihi bilgisini tutar.
- status: Öntanımlı değer 2'dir. 2, yönetici tarafından henüz onaylanmamış hesapları; 1, onaylanmış hesapları, 0 ise onaylanmamış hesapları gösterir.
- point: Kan bağışçısının sistemdeki puanı bilgisini tutar.
- city_id: Kan bağışçısının kayıtlı olduğu ilin id'sini tutar.
- district_id: Kan bağışçısının kayıtlı olduğu ilçenin id'sini tutar.
- address: Kan bağışçısının kayıtlı olduğu adres bilgisini tutar.

2. Kurum ve Kuruluşlar Tablosu: institutes

Field	Type	Null	Default	Key	Extra
id	int(4)	NO	PRI	NULL	auto_increment
name	varchar(128)	NO	PRI	NULL	
email	varchar(64)	NO		NULL	
password_digest	varchar(128)	NO		NULL	
phone_number	varchar(16)	YES		NULL	
image	varchar(32)	YES		NULL	
status	int(2)	YES		2	
role_id	int(2)	NO		NULL	
city_id	int(4)	NO		NULL	
district_id	int(4)	NO		NULL	
address	varchar(255)	YES		NULL	
created_at	datetime	NO		NULL	
updated_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- name: Kurumun ad bilgisini tutar.
- email: Kurumun email adresi bilgisini tutar.
- password_digest: Kullanıcı parolasının hashlenmiş 128 karakterlik bilgisini tutar.
- phone_number: Kurumun telefon numarası bilgisini tutar.
- image: Kullanıcı resim yolu bilgisini tutar.
- status: Öntanımlı değer 2'dir. 2, yönetici tarafından henüz onaylanmamış hesapları; 1, onaylanmış hesapları, 0 ise onaylanmamış hesapları gösterir.
- role_id: Kurumun rolüne roles tablosunda karşılık gelen id'sini tutar.
- city_id: Kurumun kayıtlı olduğu ilin id'sini tutar.
- district_id: Kurumun kayıtlı olduğu ilçenin id'sini tutar.

3. Kurum ve Kuruluş Roller Tablosu: roles

Field	Type	Null	Default	Key	Extra
id	int(2)	NO	PRI	NULL	auto_increment
name	varchar(32)	NO		NULL	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- name: Rolün ad bilgisini tutar. Bu, 'Kan Bankası', 'Hastane' veya 'Klinik' olabilir.

4. Kan Grupları Tablosu: blood_groups

Field	Type	Null	Default	Key	Extra
id	int(2)	NO	PRI	NULL	auto_increment
name	varchar(32)	NO		NULL	
created_at	datetime	NO		NULL	
updated_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- name: Kan grubunun ad bilgisini tutar.

5. Kan Bağışı Kayıt Tablosu: blood_makings

Field	Type	Null	Default	Key	Extra
id	int(8)	NO	PRI	NULL	auto_increment
donor_id	int(8)	NO		NULL	
institute_id	int(4)	NO		NULL	
blood_making_date	date	NO		NULL	
comment	varchar(255)	YES		NULL	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- donor_id: Kan bağışını yapan donörün donors tablosundaki id'sini tutar.
- institute_id: Kan bağışını alan kurumun institutes tablosundaki id'sini tutar.
- blood_making_date: Kan bağışının gerçekleştiği tarih bilgisini tutar.
- comment: Kan bağışı hakkında yorum bilgisini tutar.

6. Donör İstekleri Tablosu: donor_requests

Field	Type	Null	Default	Key	Extra
id	int(8)	NO	PRI	NULL	auto_increment
donor_id	int(8)	NO		NULL	
blood_group_id	int(2)	NO		NULL	
status	int(2)	YES		2	
content	TEXT	NO		NULL	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- donor_id: İstekte bulunan donörün donors tablosundaki id'sini tutar.
- blood_group_id: İstekte bulunan donörün kan grubunun blood_groups tablosundaki id'sini tutar.
- status: İsteğin yönetici tarafından incelenip incelenmediği bilgisini tutar.
- content: Donörün yapacağı isteğin açıklamasını tutar.

7. Donör Sorguları Tablosu: searches

Field	Type	Null	Default	Key	Extra
id	int(11)	NO	PRI	NULL	auto_increment
blood_group_id	int(2)	NO		NULL	
institute_id	int(4)	YES		0	
city_id	int(4)	YES		NULL	
district_id	int(4)	YES		NULL	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- blood_group_id: Aramadaki kan grubunun blood_groups tablosundaki id'sini tutar.
- institute_id: Aramayı yapan kurumun institutes tablosundaki id'sini tutar.
- city_id: Aramadaki şehirin cities tablosundaki id'sini tutar.
- district_id: Aramadaki ilçenin districts tablosundaki id'sini tutar.

8. Yönetici Tablosu: admins

Field	Type	Null	Default	Key	Extra
id	int(2)	NO	PRI	NULL	auto_increment
first_name	varchar(32)	NO		NULL	
last_name	varchar(32)	NO		NULL	
email	varchar(64)	NO		NULL	
password_digest	varchar(128)	NO		NULL	
image	varchar(32)	YES		NULL	
phone_number	varchar(16)	YES		NULL	
status	int(2)	YES		2	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- first_name: Yöneticinin adı bilgisini tutar.
- last_name: Yöneticinin soyadı bilgisini tutar.
- email: Yöneticinin email adresi bilgisini tutar.
- password_digest: Kullanıcı parolasının hashlenmiş 128 karakterlik bilgisini tutar.
- image: Kullanıcı resim yolu bilgisini tutar.
- phone_number: Yöneticinin telefon numarası bilgisini tutar.
- status: Yönetici seviyelerini tutar.

9. İller Tablosu: cities

Field	Type	Null	Default	Key	Extra
id	int(4)	NO	PRI	NULL	auto_increment
city	varchar(32)	NO		NULL	
plate_code	tinyint(2)	NO		NULL	
phone_code	int(4)	NO		NULL	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

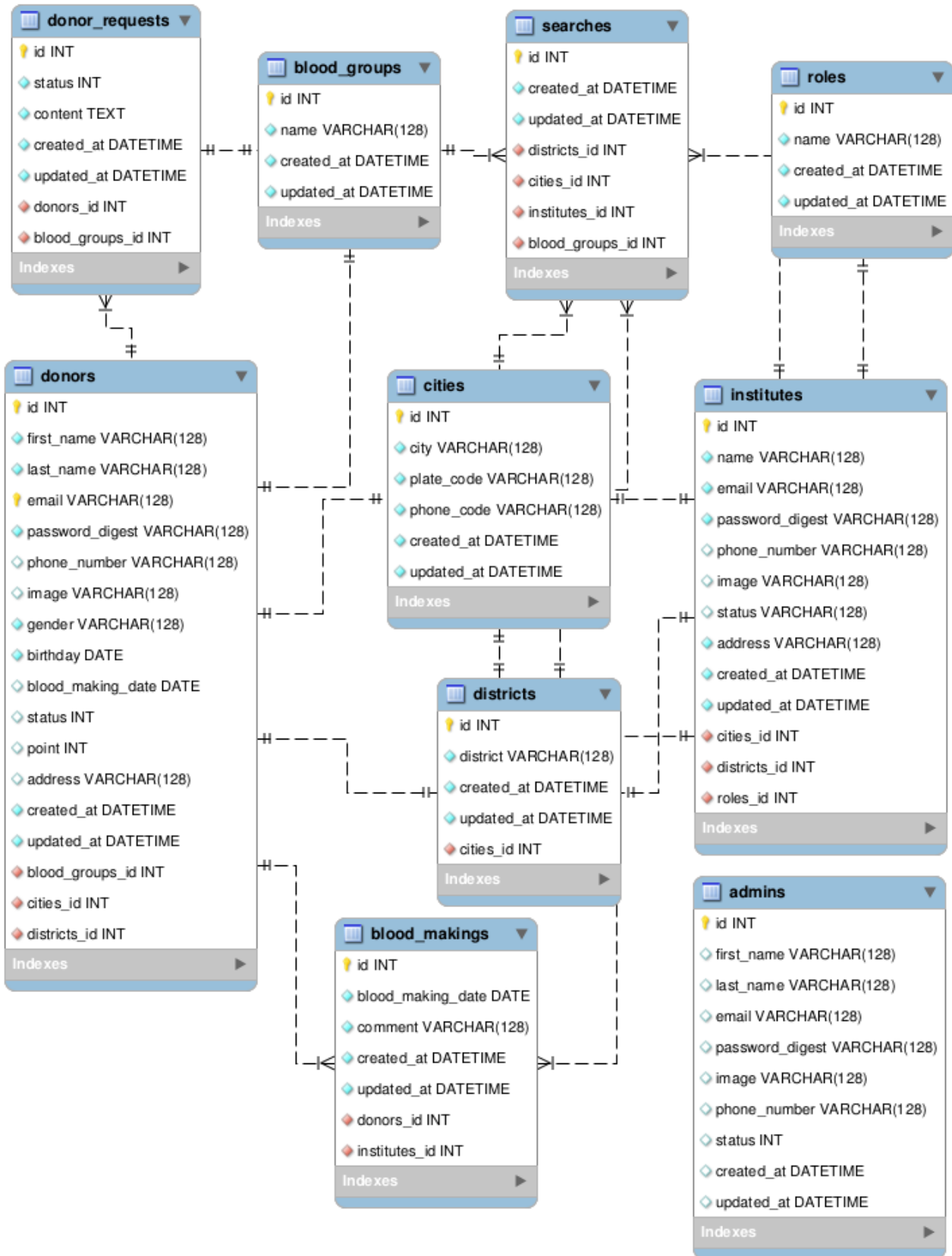
- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- city: İlın adı bilgisini tutar.
- plate_code: İlın plaka kodu bilgisini tutarç
- phone_code: İlın telefon kodu bilgisini tutar.

10. İlçeler Tablosu: districts

Field	Type	Null	Default	Key	Extra
id	int(4)	NO	PRI	NULL	auto_increment
city_id	int(4)	NO		NULL	
district	varchar(32)	NO		NULL	
created_at	datetime	NO		NULL	
update_at	datetime	NO		NULL	

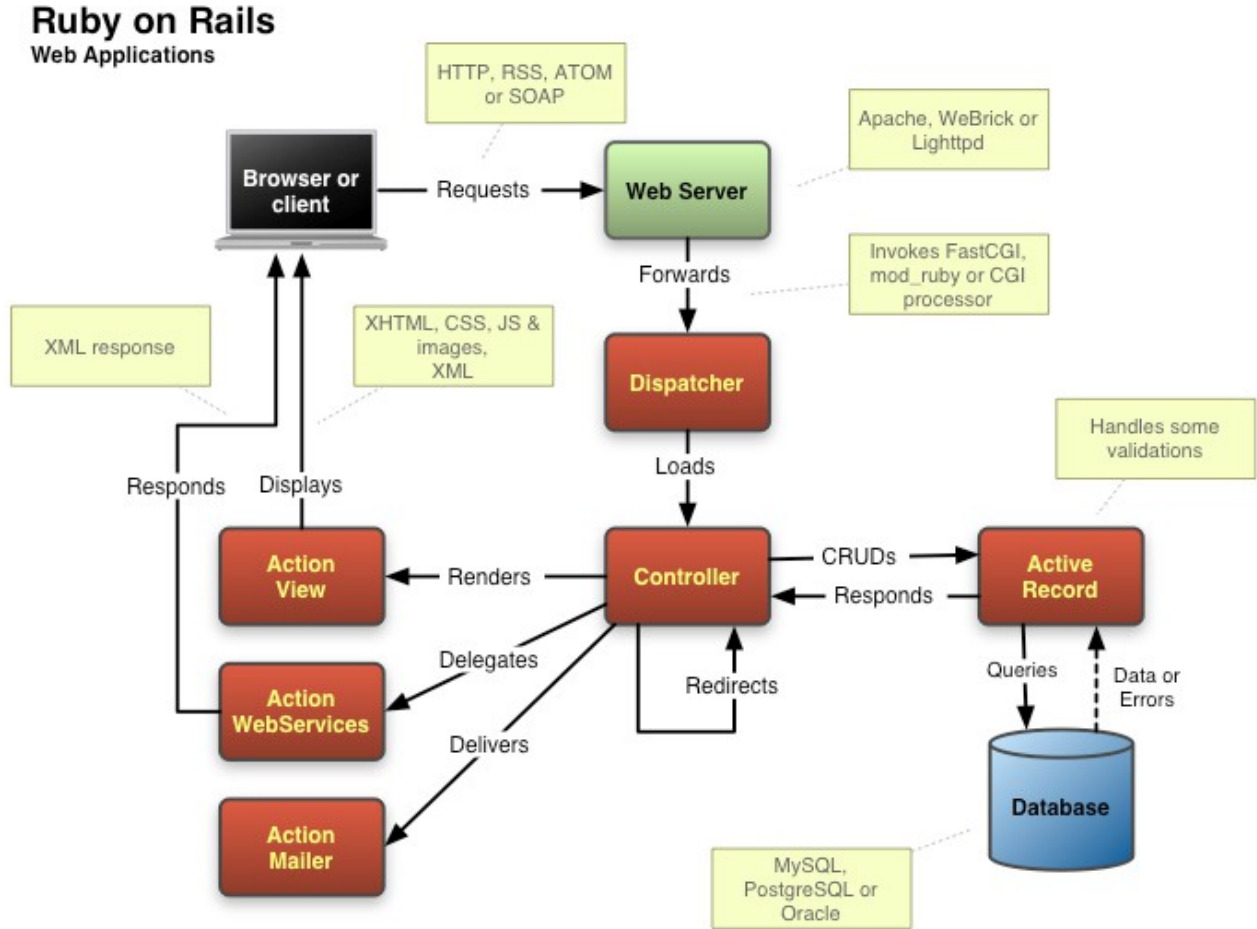
- id: Bu tablo için birincil anahtardır. Otomatik olarak oluşturulur.
- city_id: İlçenin bağlı olduğu ilin cities tablosundaki id'sini tutar.
- district: İlçenin adı bilgisini tutar.

Tablo İlişkileri

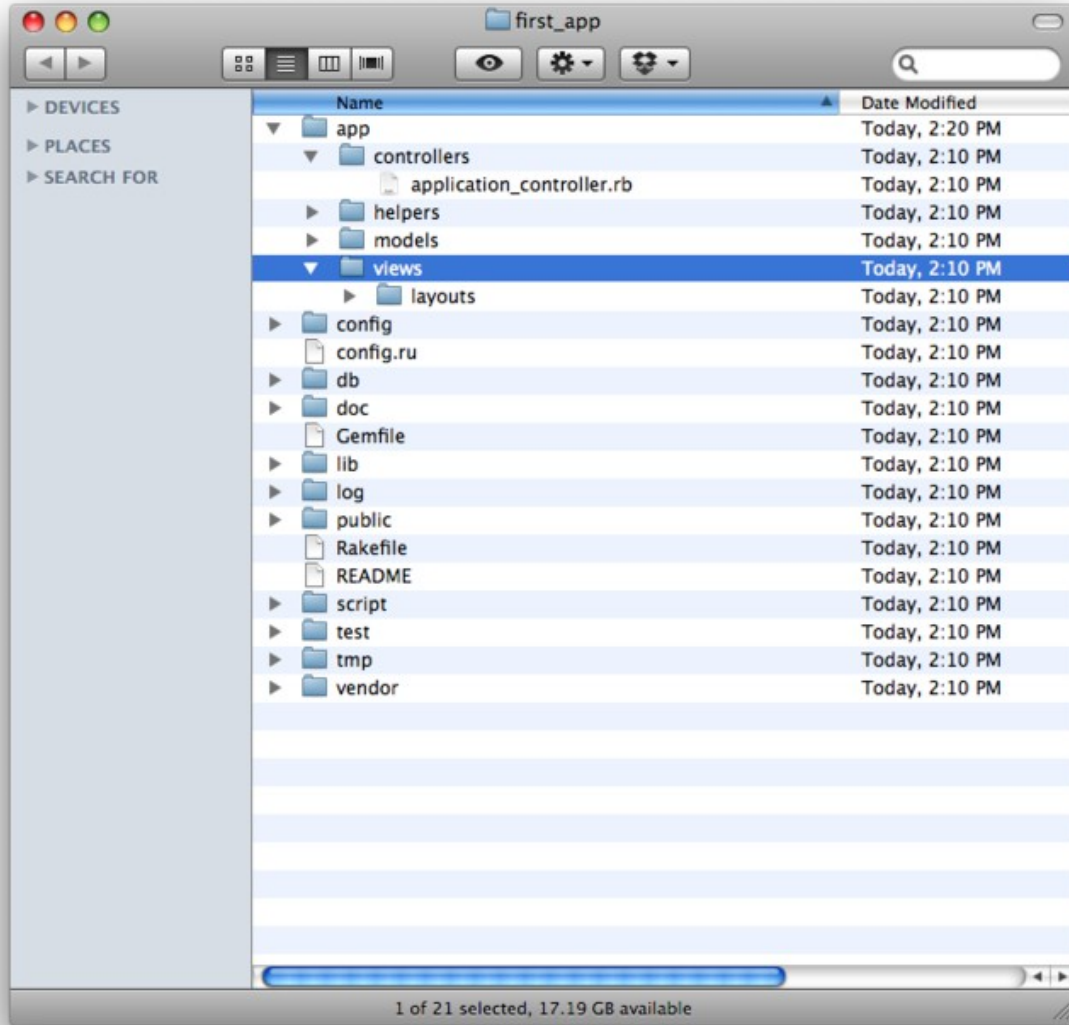


Yazılım Tasarımı

Geliştirilmeye başlanan bu sistemde herşeyden önce Ruby on Rails'ın yapısından biraz bahsetmek gerekir.



Şimdi de dizin yapısı ise (örnektir):



Rails MVC yapısı ile (yani model, view, controller) yazılımcıların daha hızlı ve az açık verecek bir web uygulaması oluşturmalarına imkan tanır.

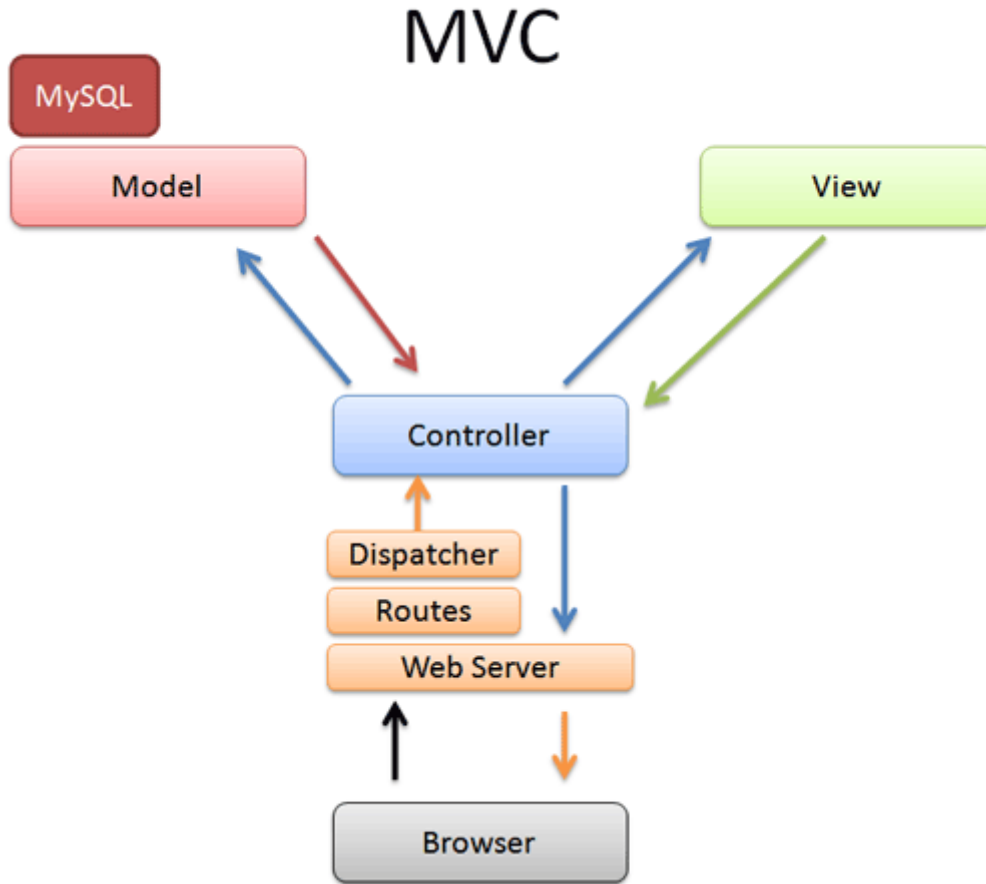
Yazılım tasarımı kısmında asıl işimiz 'app' dizini içerisinde olacaktır.

```
$ ls app/  
controllers  
  application_controller.rb  
helpers  
  application_helper.rb  
mailers  
models  
views  
  layouts  
  application.html.erb
```

Yazılımın son halinde 'app' dizini:

```
controllers
  admin
    admins_controller.rb
    donorrequests_controller.rb
    institutes_controller.rb
    sessions_controller.rb
    bloodmakings_controller.rb
    donors_controller.rb
    roles_controller.rb
  admin_controller.rb
  application_controller.rb
  home_controller.rb
  institute_controller.rb
  user_controller.rb
helpers/
  admin/
  ...
  application_helper.rb
  ...
mailers/
models/
  admin.rb
  blood_group.rb
  blood_making.rb
  city.rb
  district.rb
  donor.rb
  donor_request.rb
  institute.rb
  role.rb
  search.rb
views/
  admin/
admins/
  bloodmakings/
  donorrequests/
  donors/
  header.html.erb
  index.html.erb
  institutes/
  personal.html.erb
  roles/
  sessions/
home/
institute/
layouts/
user/
```


Anlatıma yukarıdaki dizin yapısından başlayalım. controller yapısı klasik Rails anlatımlarından biriyle daha aşağıda özetlenmiştir.



Rails routing yapısına göre örneğin anasayfadaki donör kayıt merkezi için “/home/register” isteği çalışmaktadır. Bu şu demektir:

home => controller adı
register => action adı

Yani home controller altındaki register fonksiyonunu çalıştır.
config/routers.rb de bu get metodunu belirtelim:

```
get 'home/register'
```

Web sayfamızdan gelen bu (/home/register) istekte Rails önce controller yapısından home_controller.rb'ye arkasından home controller yapısında register action'a bakmaktadır. Aslında register action dediğimiz home_controller.rb'de bulunan bir register fonksiyonudur. Rails üzerinde bu fonksiyonlar action olarak nitelendirilmektedir. Şimdi home_controller.rb dosyasına bakalım

```
class HomeController < ApplicationController
  def register
  end
end
```

Rails bu nesneden sonra views altındaki home dizininde register.html.erb'yi arayacaktır. Yani app/views/home/register.html.erb dosyası aranacaktır. Yok ise "Template#register" hatası yani varolmayan template hatası alınacaktır.

app/views/home/register.html.erb yorumlanabilen rails tarafından Actionview olarak adlandırılan bu erb dosyasına yani /home/register isteğine baktığımızda burada dinamik istekler görürüz. Kan grupları, iller ve ilçeler gibi.

Şimdi bu isteklerin karşılık bulduğu noktaları inceleyelim:

```
<select class="input-xlarge required" id="select02" name="blood_group_id">
  <option value=""></option>
  <% BloodGroup.all.each do |blood| %>
    <option value="<%= blood[:id] %>"><%= blood[:name] %>"></option>
  <% end %>
</select>

<select class="input-xlarge required" id="select-city" name="city_id">
  <option value=""></option>
  <% City.all.each do |city| %>
    <option value="<%= city[:id] %>"><%= city[:city] %>"></option>
  <% end %>
</select>
```

Gördüğümüz gibi bu istekleri bu şekilde istemci tarafına gönderiyoruz. Peki ama bu kullanım ne demek. Biraz yukarda modelleri anlatırken donors tablosundan bahsetmiştik ve bu tablonun modelinin 'Donor' olduğunu söylemiştik. Yine aynı şekilde veritabanımızda cities (dikkat edelim çoğul şekilde) tablomuz var ve bu tablonun model adı yani kullanabileceğimiz nesnesinin adı City'dir (tekil ve büyük harfle başlıyor).

Model kullanımına biraz değinecek olursak: (Model => kullandığımız tablonun model adı)

```
Model.all => tablodaki tüm bilgileri getirir.
Model.find(1) => tablodaki 1 id'li kayıtlı getirir.
Model.first => tablonun birinci elamanını getir
Model.last => tablonun sonuncu elamanını getir
Model.find(:all, :conditions => {:status => 2}) => tablodaki status sütunları 2 olan kayıtları getir.
Model.find(:first, :conditions => {:status => 2}) => tablodaki status sütunları 2 olan ilk kayıtlı getir.
```

gibi...

Yukarıda City örneğinde de gördüğümüz gibi City.all diyerek tablodaki tüm kayıtları getirdik. ve bu kayıtlar üzerinde gezerek bir option'lar üzerinden id ve adını yazarak select div'ini oluşturduk. (Bakınız veritabanı tasarımı bölümü cities tablosu)

Kan grupları da yine aynı şekilde: veritabanı tasarımı bölümünde anlatmıştık. blood_groups tablomuzun adı ve dolayısıyla da modelimizin adı BloodGroup (tekil ve büyük harfle başlıyor). Burada özel olarak veritabanı modellemizde '_' olarak ayrılmış isimle tekrar ikinci büyük harfe dönüşüyor ve kullanılıyor.

Örneğin:

Veritabanı Tablo Adı	Model Adı	Görevi
donor_requests	DonorRequest	Donör (Kan Bağışçısı) İstekleri
blood_groups	BloodGroup	Kan grupları
blood_makings	BloodMaking	Kan verme kayıtları

Kullanıcı isteklerine model, sql ve actionview ile bu şekilde cevap verebiliyoruz.

Peki bu kaydı yorumlayan kod nerede?

Öncelikle register.html.erb'ye bakalım burada kayıt işlemlerini gerçekleştireceğimiz form tag'ının action adı '/home/register_save' olarak görünüyor.

```
<form action="/home/register_save" method="post" enctype="multipart/form-data" class="form-horizontal" >
```

Hemen config/routers.rb'de bunun bir post methodu olduğunu belirtelim.

```
get 'home/register'
post 'home/register_save'
```

Nasıl get methodunda home controller nesnesinde register fonksiyonu çalışıyor ise şimdi de register_save fonksiyonu çalışacak.

register_save fonksiyonuna bakalım:

```
def register_save
  if params[:password] == params[:password2]
    donor = Donor.new({
      :tc => params[:tc],
      :first_name => params[:first_name],
      :last_name => params[:last_name],
      :blood_group_id => params[:blood_group_id],
      :gender => params[:gender],
      :birthday => params[:birthday],
      :email => params[:email],
      :password => params[:password],
      :phone_number => params[:phone_number],
      :city_id => params[:city_id],
      :district_id => params[:district_id],
      :address => params[:address],
    })
```

```

if donor.save
  if params[:image] and response = Image.upload('donors', donor[:id], params[:image])
    donor[:image] = response[1] if response[0]
    donor.save
  end
  flash[:notice] = "Kaydınız Alınmıştır. Teşekkür Ederiz"
else
  flash[:error] = "Kaydınız Alınamadı"
end
else
  flash[:error] = "Parolalar Eşleşmiyor"
end
end
redirect_to '/home/register'
end

```

Burada herşeyden önce gelen parolaların eşitliğine bakılıyor. Arkasından requestlerden (ruby'de bunlar params değişkenleri olarak adlandırılır) gelen verileri Donor modeline gönderiyoruz eğer kayıt başarılıysa ve resim isteği varsa bu isteğe cevap ver. Resim kaydı başarılı ise resmin sadece adını bu kayda ekle. Son olarak başarılı veya başarısız bir sonuç döersen bunları flash'a ver. flash değişkeni bunları nasıl görüntüleyeceğini biliyor. Nasıl mı? Hemen bakalım framework'umuzun layout dosyasına bakalım app/views/layouts/application.html.erb

```

<% [:notice, :error, :alert].each do |level| %>
  <% unless flash[level].blank? %>
    <div data-alert="alert" class="alert alert-<%= level %> fade in">
      <a class="close" data-dismiss="alert" href="#">&times;</a>
      <%= content_tag :p, simple_format(flash[level]) %>
    </div>
  <% end %>
<% end %>

```

Eğer ki hata ve başarı ve bilgilendirme değişkenleri varsa bunları flash değişkeninden al ve görüntüle. flash değişkenini tüm sistemde bu şekilde kullanabiliyoruz.

Tüm kayıtlarımız bu şekilde ilerlemektedir.