T.C

FIRAT ÜNİVERSİTESİ MÜHENDİSLİK FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

DİETFİT MOBİL UYGULAMA

Yazılım Proje Yönetimi

Dersi Projesi

Hazırlayanlar

15290043-Fatma OĞUZ (Proje Yöneticisi)

15290059-Şeyma ALTUNBAŞ

15290021- Seren BERBER

16290032-Muhammet Yasir ADAMOĞLU

Proje Sorumlusu: Dr. Öğr. Üyesi Mustafa ULAŞ

ELAZIĞ, 2020

ÖZET

Akıllı telefonların ortaya çıkması ve yaygınlaşmasıyla beraber, kullanıcılar hakkında veri izleyen ve kaydeden birçok mobil uygulama geliştirilmiştir. Sağlıklı ve nitelikli bilgiye hızlı ve direkt ulaşmayı sağlayan bu uygulamalar aynı zamanda düzenli beslenme ve kaliteli bir yaşam için diyetisyenlere erişim imkanı da sunmaktadır. Fakat yapılan araştırmalarda kullanıcının profesyonel olarak sağlıklı ve dengeli beslenme problemi için dijital dönüşüme uyum sağlayan yeterli bir uygulama bulunmadığı fark edilmiştir. Modern hayatın yoğun temposu içinde sağlıklı ve dengeli beslenmeye yeterli zaman ayrılmadığı için beslenme alışkanlıklarımızda olumsuz olarak etkilenmektedir. Türkiye'de artan obezite prevelansı göz önünde bulundurarak obezitenin yanında getirdiği metabolik rahatsızlıklarla toplumun sağlığının kötüye gidişini düzeltmek ve bu doğrultuda sağlık sektöründe yapılan maddi ve manevi harcamaların önüne geçerek ülkemizin kalkınmasında etkili olabilecek bir uygulama geliştirilmesi amaçlanmaktadır.

İÇİNDEKİLER

1. GİRİŞ

1.1 Projenin Amacı

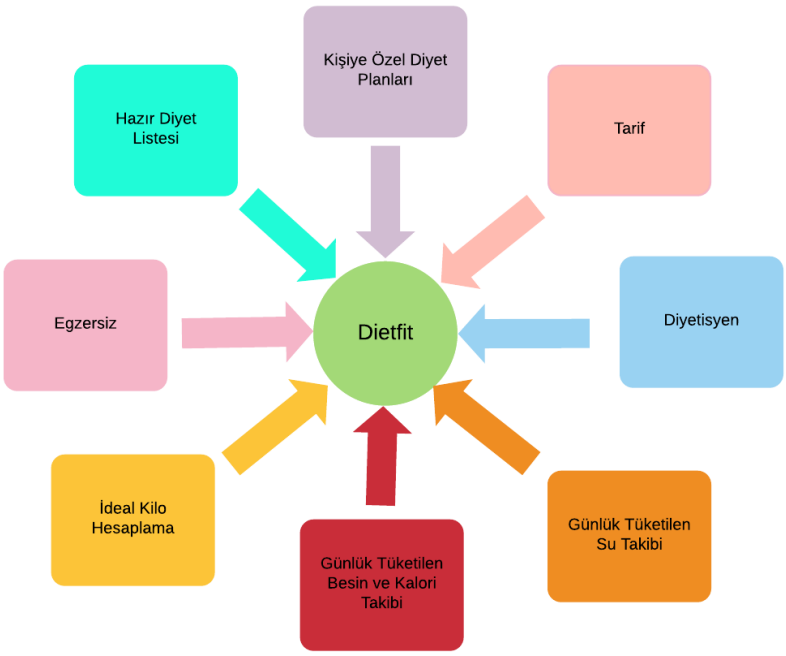
Mekana bağlı olmaksızın her gün çeşitli nedenler için kullandığımız ve artık hayatımızın vazgeçilmez unsurları olan akıllı cihazlar toplumun sağlıklı ve düzenli beslenmesi gibi alanlarda da kullanılmaya başlanmıştır. İnsanların sağlıklı ve düzenli beslenmelerine yardımcı olarak istediği zaman, istediği yerde, günlük tükettiği besinlerin ve su takibinin sağlandığı, hazır ve diyetisyen tarafından oluşturulmuş diyet listeleri ile tarif, egzersiz ve ideal kilo hesaplama gibi birçok özelliği kapsayacak bir mobil uygulama geliştirilmesi amaçlanmaktadır.

1.2 Projenin Kapsamı

Dietfit; Android ve Ios ortam için geliştirilecek olup, yaş veya cinsiyet gözetmeksizin her kesimi kapsayacaktır.

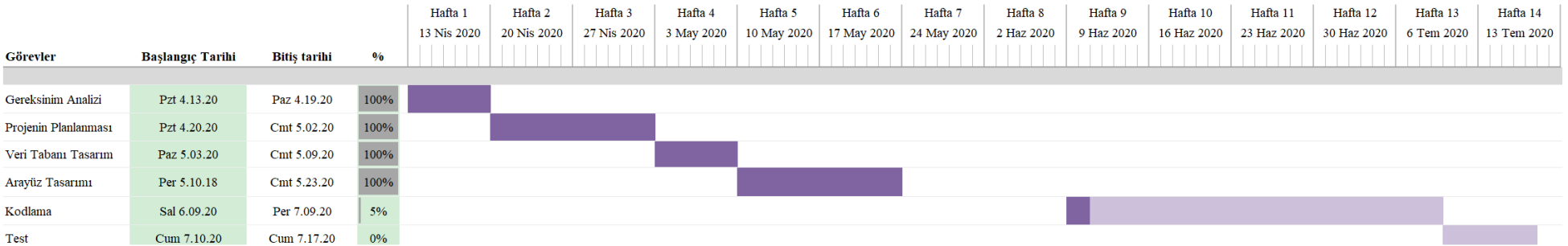
2. PROJE PLANI

Dietfit; insanların beslenme alışkanlıklarını düzenlemek amacıyla günlük tükettiği besin ve su miktarının takibini sağlayan, hazır ve diyetisyen tarafından oluşturulmuş diyet listeleri ile kilo kontrolüne yardımcı olan, tarifler kısmı ile kullanıcının öğünlerini zenginleştiren ve spor salonuna gitmeye gerek kalmadan aylık egzersiz planlarına erişebilinen bir mobil uygulamadır.



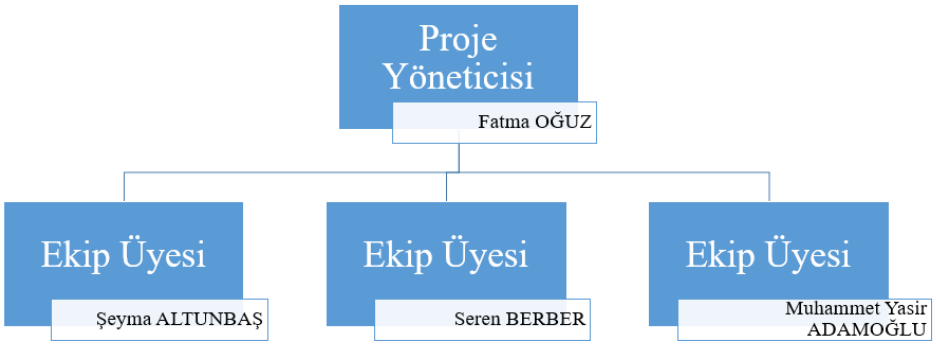
Şekil 2.1: Uygulamanın Genel Yapısı

2.1 Proje Zaman-İş Planı



Şekil 2.2: Proje İş- Zaman Çizelgesi.

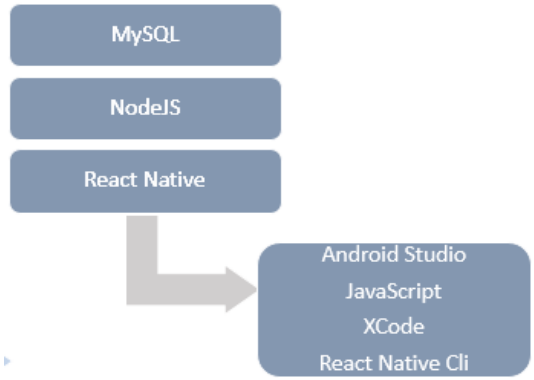
2.2 Proje Ekip Yapısı



Şekil 2.3: Proje Ekibi.

2.3 Önerilen Sistemin Teknik Tanımları

Veri tabanı için MySQL kullanılmıştır ve proje gerçekleştirimi React Native ile gerçekleştirilecektir. React Native için gerekli programlama araçlarına aşağıdaki şekilde yer verilmiştir.



Şekil 2.4: Kullanılan Teknik Araçlar

2.4 Kullanılan Özel Geliştirme Araçları ve Ortamları

Proje Yönetim Araçları	Takım Yönetim Araçları	Çözümleme ve Tasarım Araçları
Trello Github	Slack	Creately Lucidchart StarUML Adobe Photopshop CS6 Balsamiq

Programlama Araçları	Sınama Araçları	Destek Araçları
MySQL NodeJS Android Studio XCode React Native Cli	Android iOS	Google Chrome Internet Explorer Safari Mozillla Firefox

Şekil 2.5: Kullanılan Geliştirme Araç ve Ortamları.

2.5 Proje Standartları, Yöntem ve Metodolojiler

Aşama	Kullanılan Yöntem/Araçlar	Ne İçin Kullanıldığı	Çıktı
Planlama	- Veri Akış Şemaları - Süreç Belirtimleri - Görüşme - Proje Yönetim Araçları	- Süreç İnceleme - Kaynak Kestirimi - Proje Yönetimi	Proje Planı
Çözümleme	- Süreç Belirtimleri - Veri Akış Şemaları - Görüşme - Nesne-İlişki Şemaları - Veri Sözlüğü	- Süreç Çözümleme - Veri Çözümleme	Sistem Çözümleme Raporu
Çözümlemeden Tasarıma Geçiş	- Akışa Dayalı Çözümleme - Süreç Belirtimlerinin Program Tasarım Diline Dönüştürülmesi -Nesne-İlişki Şemalarının Veri Tablolarına Dönüştürülmesi	- Başlangıç Tasarım - Ayrıntılı Tasarım - Başlangıç Veri Tasarımı	Başlangıç Tasarım Raporu
Tasarım	- Yapısal Şemalar - Program Tasarım Dili - Veri Tabanı Tabloları - Veri Sözlüğü	- Genel Tasarım - Ayrıntılı Tasarım - Veri Tasarımı	Sistem Tasarım Raporu

Şekil 2.6: Proje Aşamaları.

Proje standartları yukarıda verildiği gibidir.

2.6 Kalite Sağlama Planı

Şekil 2.7: Kalite Sağlama Planı.

Projedeki kalite sağlama planımız yukarıda belirtildiği gibi olup;

Ekonomi: Yazılım uygun fiyata ve sürekli çalışabilirdir.

Tamlık: Projede herhangi bir açık bulunmamalı ve tüm fonksiyonlar çalışır ve tam olmalıdır.

Yeniden Kullanılabilirlik: Yazılım kullanıcı gereksinimlerine göre yeniden uyarlanabilirdir.

Etkinlik: Kullanıcı sistemi etkin bir biçimde kullanacaktır.

Bütünlük: Veriler ve işlemler arasındaki tutarlılık korunacaktır.

Güvenilirlik: Yazılım güncel bir sürüme sahip olup hatalı girdilere ve kullanıcı yanlışlıklarına karşı korumalı olacaktır.

Modülerlik: Kullanıcı uygulamada her sayfada söz sahibi olacaktır.

Belgeleme: Sistemi özetleyen bir doküman oluşturulmuştur.

Kullanılabilirlik: Uygulama genel bir kitleye hitap ettiğinden kullanıcı arayüzü basit oluşturulmuş, karmaşık sistemlerden kaçınılmıştır.

Değiştirilebilirlik: Veri tabanına erişim yetkisi olan (admin, diyetisyen) kişiler sistemde değişiklik yapabilecektir.

Esneklik: Proje farklı platformlar üzerinde çalışacağından esnektir.

Genellik: Proje cinsiyet ve yaş farkı gözetmeksizin her kesim tarafından kullanılabileceğinden geneldir.

Sınanabilirlik: Proje sınanabilir olacaktır.

Taşınabilirlik: Sistem Android ve Ios cihazlarda taşınabilir ve kullanılabilirdir.

Birlikte Çalışabilirlik: Sistemin tamamı veya büyük bir kısmı başka bir sistemle çalışabilir olacaktır.

2.7 Konfigürasyon Yönetim Planı

Sistemin yapısındaki bazı bileşenlerin değişmesi sonucu güncelliğin kaybedilmesi durumunda hazırlanan konfigürasyon planı uygulanmalıdır.

- Sistemde istenmeyen herhangi bir durum olması
- Herhangi bir sebepten dolayı sistemden ayrılan diyetisyen olması
- Abonelik iptali

gibi durumlar için konfigürasyon yönetim planı oluşturulmuştur.

2.8 Kaynak Yönetim Planı

Kaynak yönetiminde şu hususlar dikkate alınacaktır;

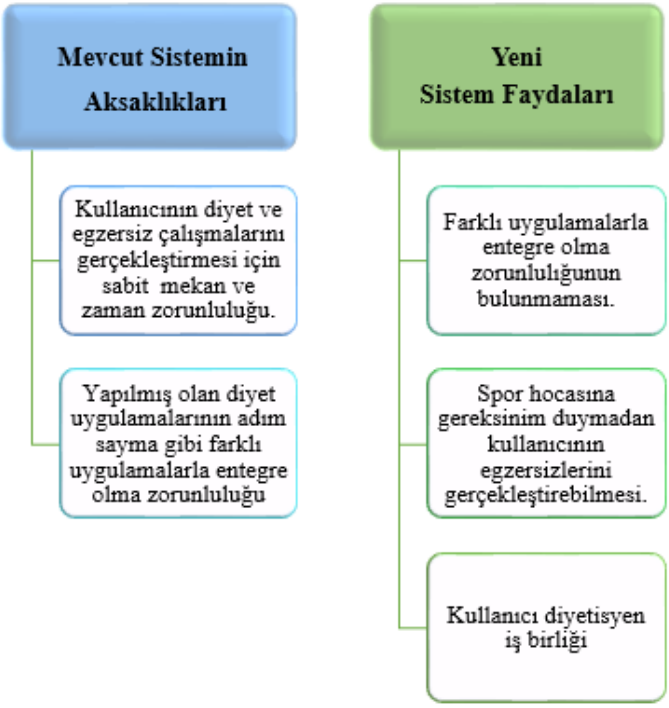
- Kalite beklendiği gibi mi?
- Yeterli hata bulunuyor mu?
- Bu projede yeni bir süreç iyileştirmesi kullanılıyor mu?
- Proje elemanları yeni bir eğitim aldı mı?
- Yeni bir araç kullanılıyor mu?
- Yeterli vakit harcanıyor mu?

2.9 Eğitim Planı

Diyetisyenlere diyet planlarını oluşturabilmeleri için 2 günlük web panel kullanım eğitimi verilecektir.

3. SİSTEM ÇÖZÜMLEME

3.1 Mevcut Sistemin İncelenmesi



Şekil 3.1: Mevcut Sistemin İncelenmesi.

3.1.1 Örgüt Yapısı

Kullanıcı; mobil cihazı ile hazır ve diyetisyen tarafından oluşturulmuş diyet listeleri, diyetisyenler, tarifler, egzersizler, ideal kilo hesaplaması ve besin ve su tüketiminin izlenilmesi etkinliklerine erişim sağlayabilir, veri girişi yapabilir.

3.1.2 İşlevsel Model

Şekil 3.2: Temel Görevlendirme

3.1.3 Varolan Yazılım/Donanım Kaynakları

Yazılım

- Sunum mantığı
- Uygulama mantığı
- Veri erişim mantığı
- Veri depolama

Donanım

- İstemci bilgisayar
- Sunucular
- Ağ

3.1.4 Varolan Sistemin Değerlendirilmesi

Dietfit ile kullanıcı diyetisyene gitmek yerine uygulama üzerinden diyetisyen ile iletişim kurarak kişiye özel diyet planı talep eder. Aynı zamanda spor hocasına gerek duymadan birçok egzersiz hareketine erişim sağlar.

3.2 Gereksenen Sistemin Modeli

Oluşturulacak uygulama da kullanıcılar mobil cihazları ile:

- İdeal kilo hesaplaması yapabilir.
- Premium üyelik satın alıp diyetisyenle iletişim kurarak kişiye özel diyet planı alabilir.
- Egzersiz çalışmalarını gerçekleştirebilir.
- Hazır diyet listelerini ve tarifleri görüntüleyebilir.
- Günlük tüketilen besin ve kalori takibini gerçekleştirebilir.
- Günlük tüketilen su takibini gerçekleştirebilir.

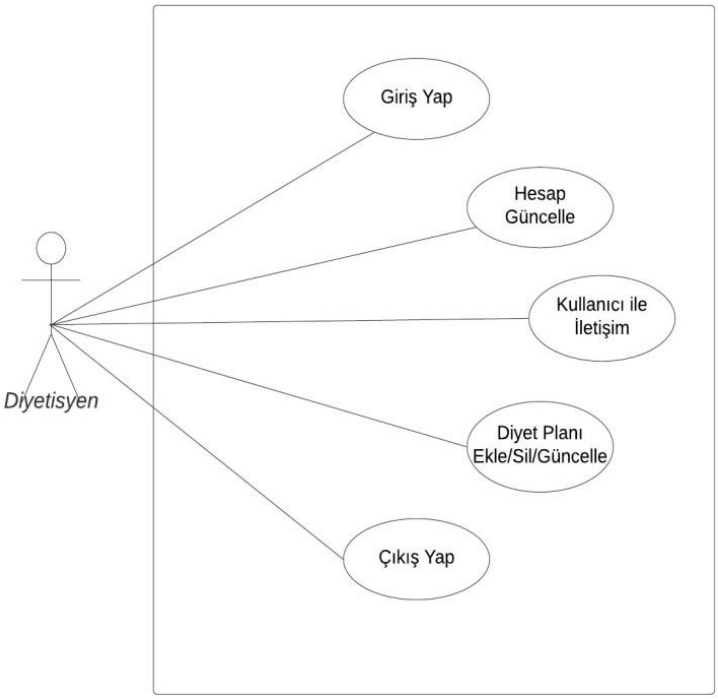
3.2.1 Use Case Diyagramı

Bir use case örneği, bir çıktı sonucu üretmek için gerçekleştirilen bir dizi etkinliği gösterir. Her kullanım durumu, harici bir kullanıcının sistemin yanıt vermesi gereken bir olayı nasıl tetiklediğini açıklar.

Aktör 1: Sistemin ana aktörü kullanıcıdır. Kullanıcı sisteme giriş yaparak diyetisyen ile iletişim kurabilir, diyet planı talep edebilir, öğünlerini yönetebilir, egzersizlere erişebilir, hazır diyet listesi ve tarif görüntüleyebilir.

Şekil 3.3: Kullanıcı Use Case Diyagramı.

Aktör 2: İkincil aktör diyetisyendir. Diyetisyen sisteme giriş yapabilir, hesabını güncelleyebilir, kullanıcı ile iletişim kurabilir, diyet planı yönetebilir.



Şekil 3.4: Diyetisyen Use Case Diyagramı.

Aktör 3: Üçüncü aktör sistemin yöneticisidir. Sistem yöneticisi tarafından yapılan eylemler şunlardır:

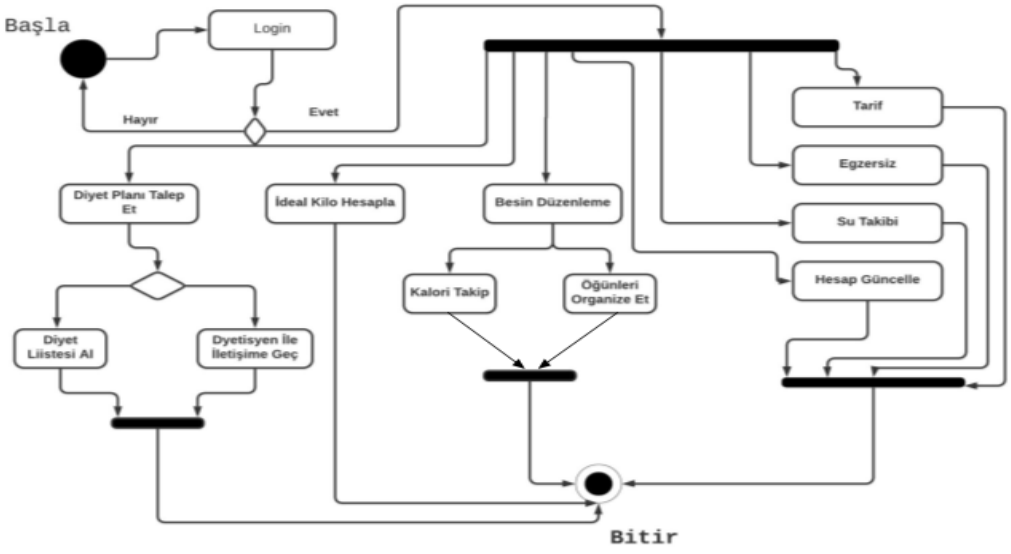
- Giriş yapabilir.
- Kullanıcı, diyetisyen, ücret ve rapor yönetimi gerçekleştirebilir.

- Diyet listesi, egzersiz, besin, tarif, öğün ve bildirim ekleyip silip güncelleyebilir.

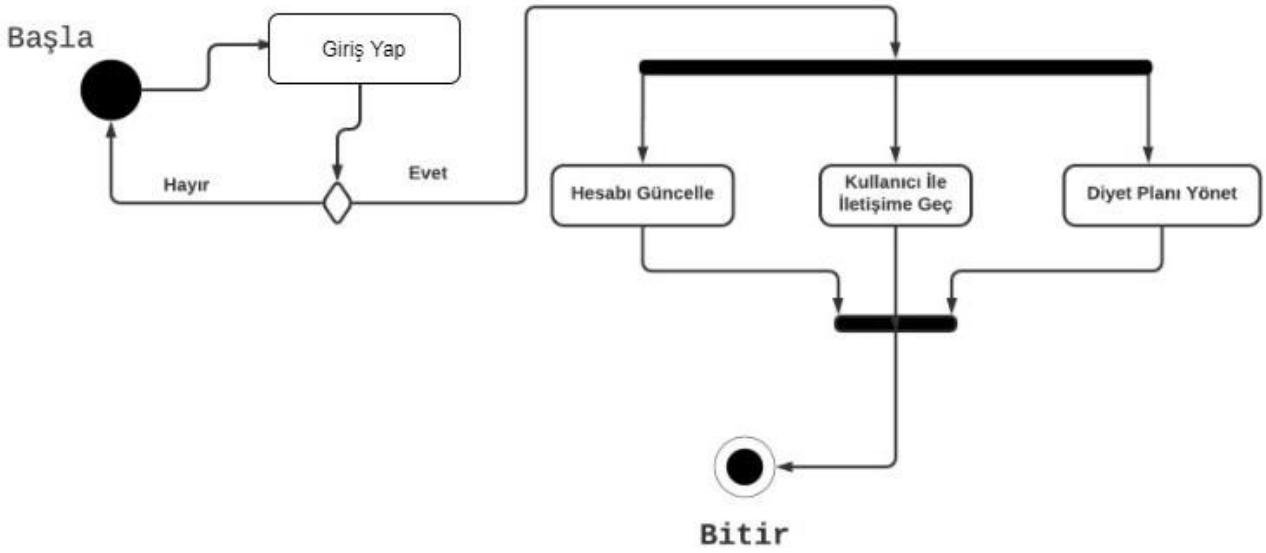
Şekil 3.5: Admin Use Case Diyagramı.

3.2.2 Activity Diyagramı

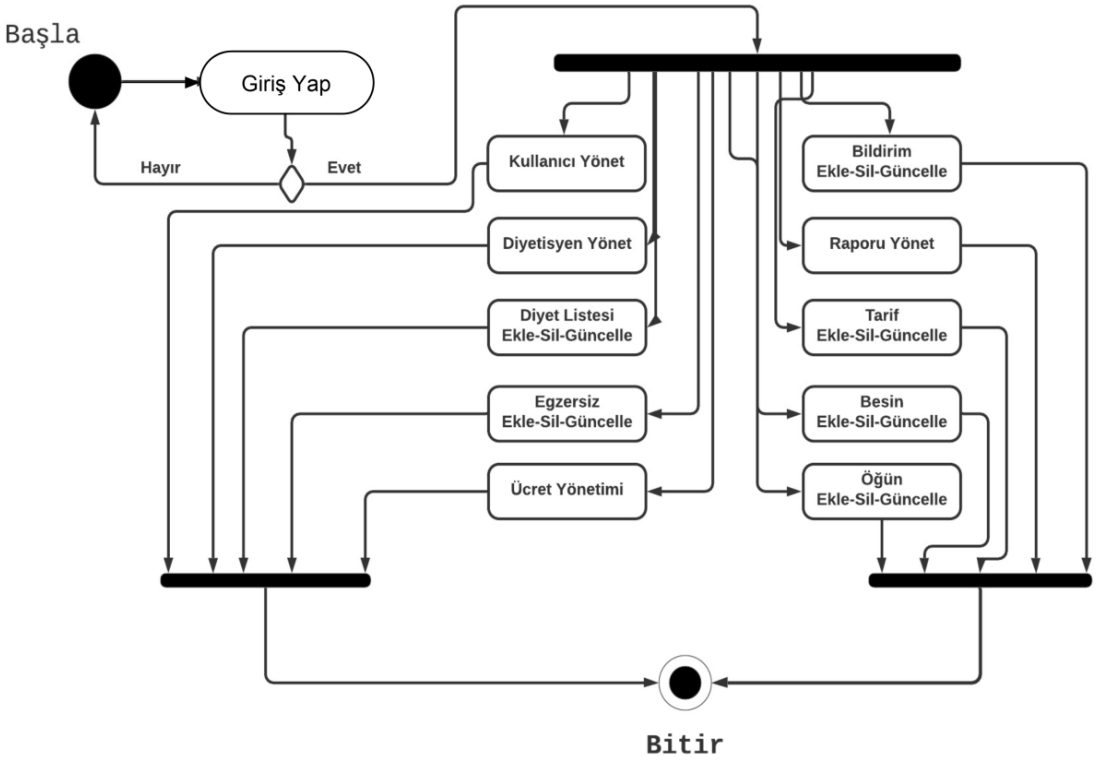
Activity diyagramı, sisteme genel bir bakış açısı sağlar ve teknik bilgisi olmayan kişilerin sistemi anlayabilmeleri için en iyi araçtır.



Şekil 3.6: Kullanıcı Activity Diyagramı.



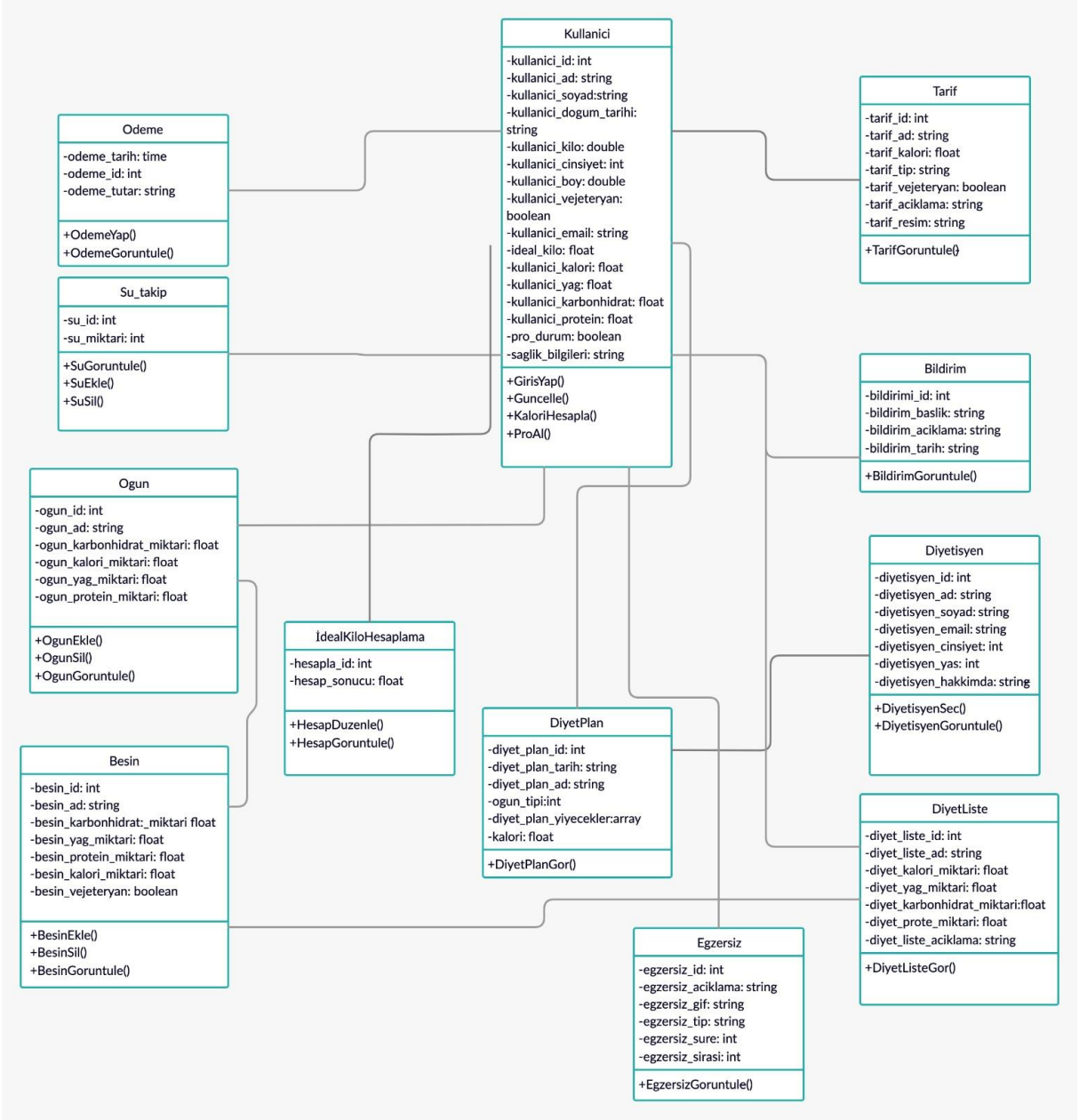
Şekil 3.7: Diyetisyen Activity Diyagramı.



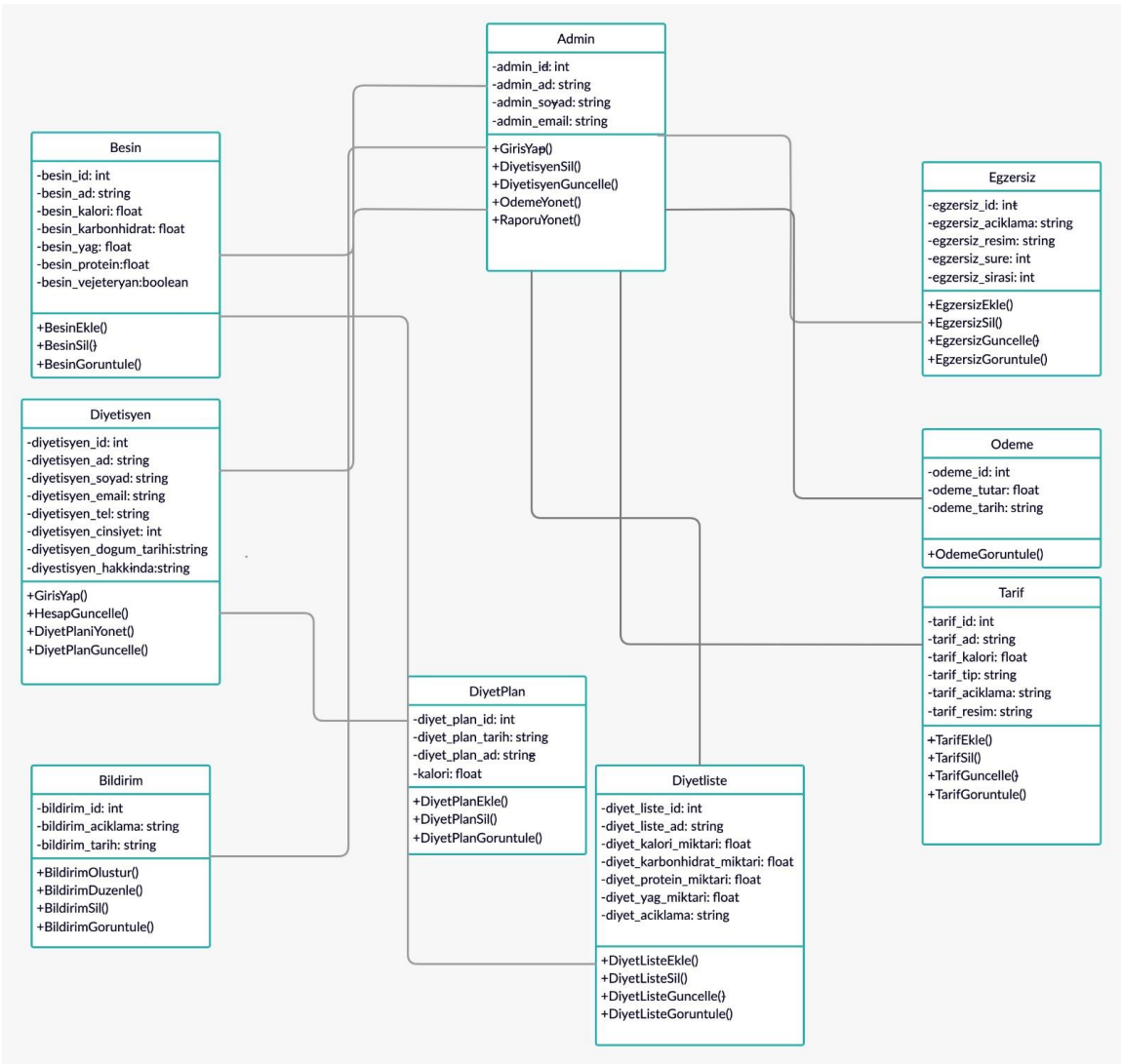
Şekil 3.8: Admin Activity Diyagramı.

3.2.3 Class Diyagramı

Sistemin sınıflarını**,** özelliklerini, işlemlerini(veya yöntemlerini) ve nesneler arasındaki ilişkileri göstererek sistemin yapısını tanımlayan statik yapılı diyagram türüdür.

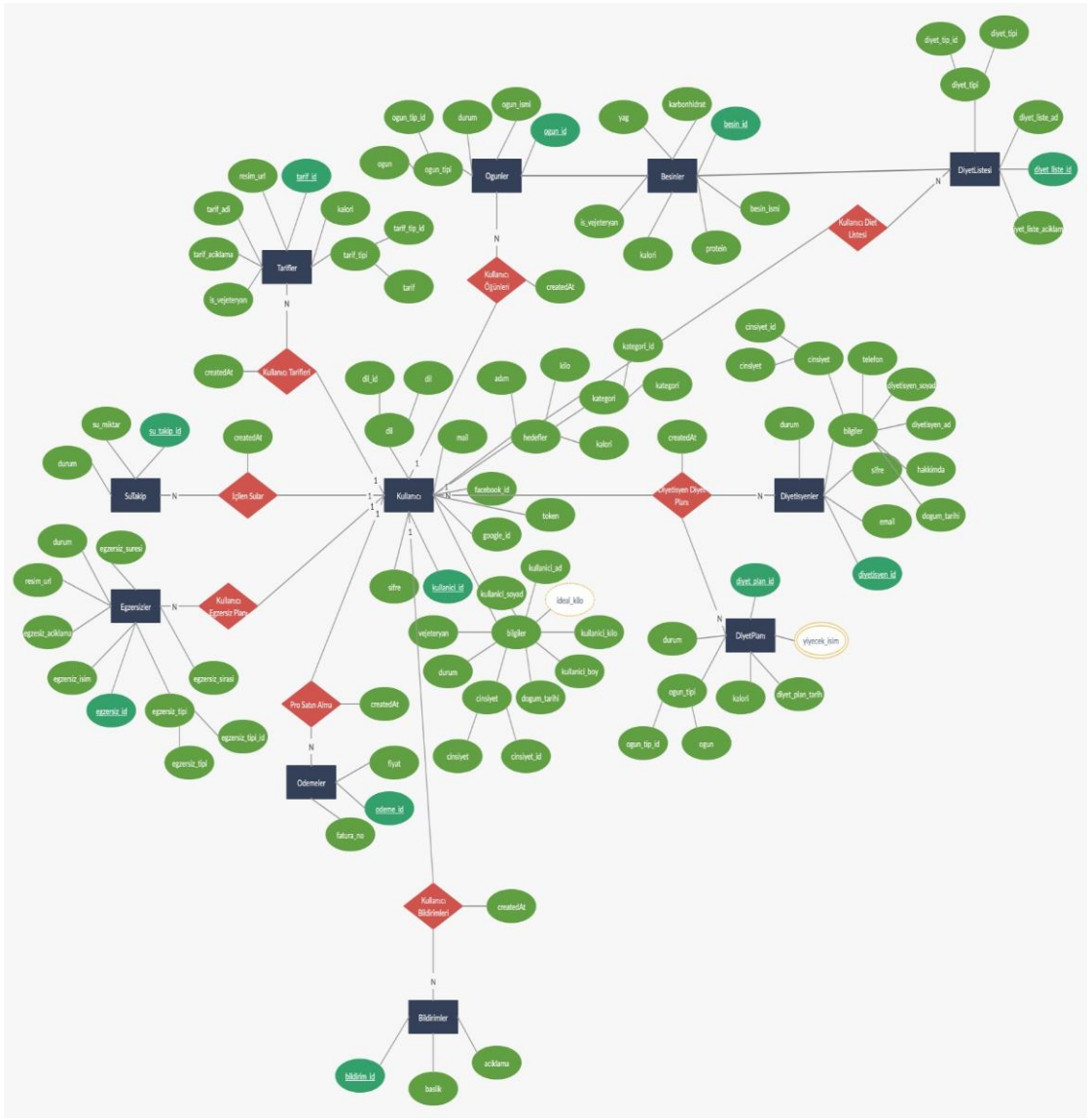


Şekil 3.9: Frontend Class Diyagramı.



Şekil 3.10: Backend Class Diyagramı

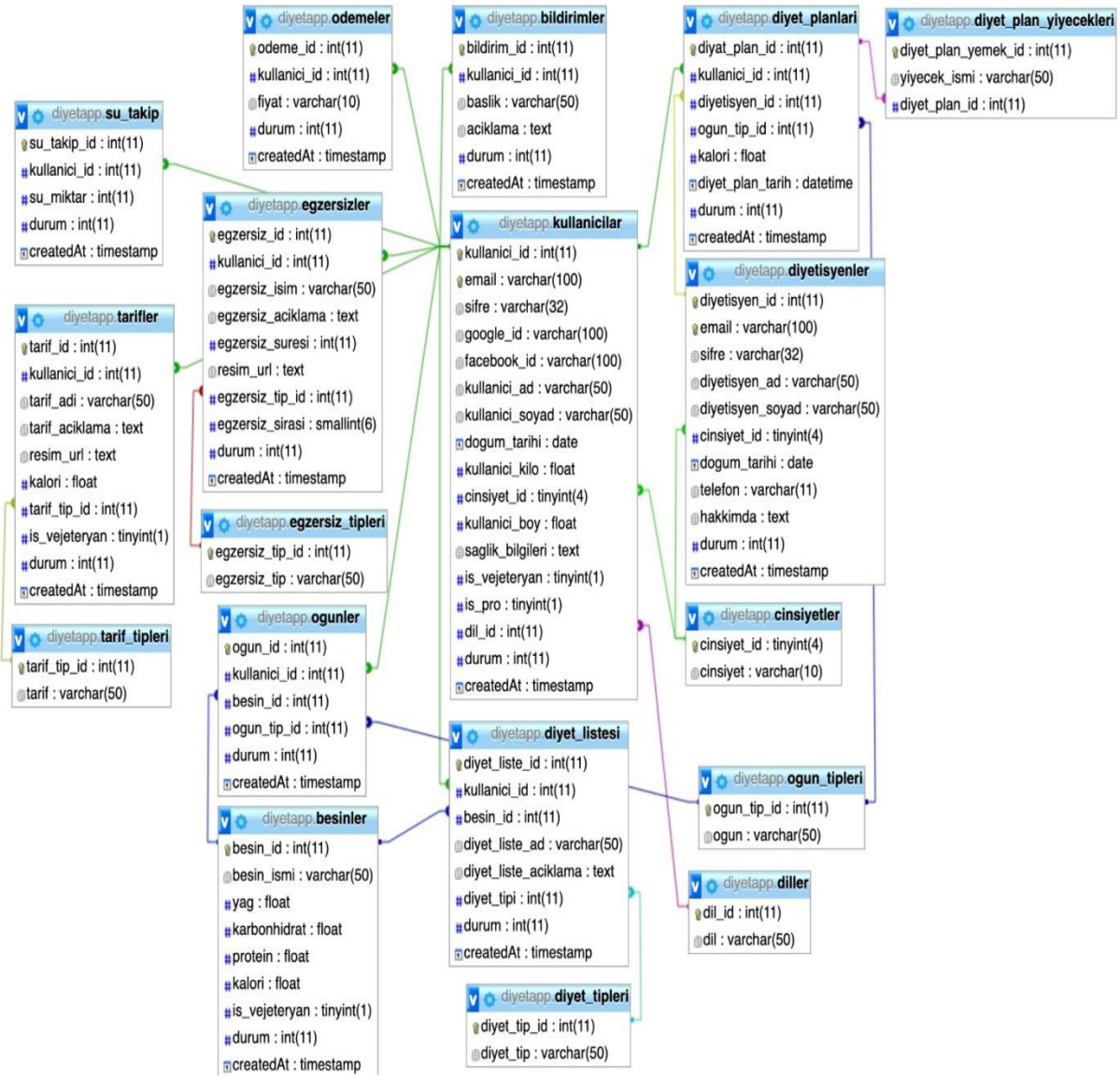
3.2.4 Varlık İlişki Diyagramı(ERD)



Şekil 3.11: ER Diyagramı

3.2.5 Veri Modeli

İlişkisel veri tabanı veri modelinde veriler tablolar üzerinden kurulan ilişkiye dayanmaktadır.



Şekil 3.12: Veri Modeli

3.2.6 Veri Sözlüğü

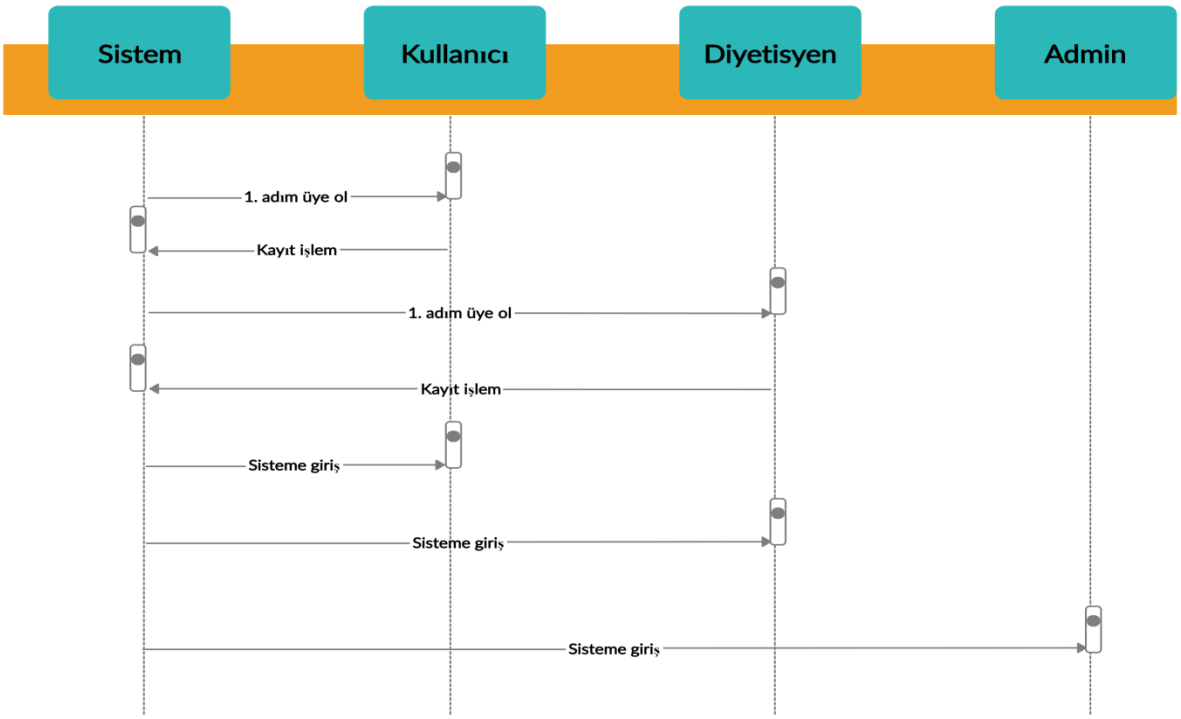
Tablo Adı	Değişken Adı	Veri Tipi	Uzunluk	Açıklama
besinler	besin_id	int	11	İlgili olaya referans.
	besin_ismi	varchar	50	Besin isim bilgisi.
	yag	int	11	Besin yağ miktarı bilgisi.
	karbonhidrat	int	11	Besin karbonhidrat miktarı bilgisi.
	protein	int	11	Besin protein miktarı bilgisi.
	kalori	int	11	Besin kalori miktarı bilgisi.
	is_vejeteryan	tinyint	1	Vejeteryanlık olup olmama bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
bildirimler	bildirim_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	baslik	varchar	50	Bildirim başlık bilgisi.
	aciklama	text		Bildirim açıklama bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
cinsiyet	cinsiyet_id	int	10	İlgili olaya referans.
	cinsiyet	varchar	6	Kadın ve erkek girilebilir.
diller	dil_id	int	11	İlgili olaya referans.
	dil	varchar	50	Ülkelere ait dil bilgisi.
diyetisyenler	diyetisyen_id	int	11	İlgili olaya referans.
	email	varchar	11	Diyetisyene ait email bilgisi.
	sifre	varchar	32	Diyetisyene ait şifre bilgisi.
	diyetisyen_ad	varchar	50	Diyetisyen isim bilgisi.
	diyetisyen_soyad	varchar	50	Diyetisyen soyad bilgisi.
	cinsiyet_id	int	11	İlgili olaya referans.
	dogum_tarihi	date		Diyetsiyen doğum tarihi bilgisi.
	telefon	varchar	11	Diyetisyen iletişim numarası bilgisi.
	hakkinda	text		Diyetisyenin hakkındaki bilgiler.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
diyet_listesi	diyet_liste_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	besin_id	int	11	İlgili olaya referans.
	diyet_liste_ad	varchar	50	Diyet listesi isim bilgisi.
	diyet_liste_aciklama	text		Oluşturulan diyet listesinin açıklama bilgisi.
	diyet_tipi_id	int	11	Oluşturulan diyet listesi tip bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
diyet_planlari	diyet_plan_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	diyetisyen_id	int	11	İlgili olaya referans.
	ogun_tip_id	int	11	İlgili olaya referans.
	kalori	int	11	Oluşturulan diyet planı kalori miktarı bilgisi.
	diyet_plan_tarih	datetime		Oluşturulan diyet planı tarih bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
diyet_plan_yiyecekleri	diyet_plan_yemek_id	int	11	İlgili olaya referans.
	yiyecek_ismi	varchar	50	Diyet planına eklenecek yiyecek bilgisi.
	diyet_plan_id	int	11	İlgili olaya referans.
diyet_tipleri	diyet_tip_id	int	11	İlgili olaya referans.
	diyet_tip	varchar	50	Oluşturulacak diyet listesi için diyet tip bilgisi.

egzersizler	egzersiz_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	egzersiz_isim	varchar	50	Egzersiz isim bilgisi.
	egzersiz_aciklama	text		Egzersiz çalışmalarının detaylı bilgisi.
	egzersiz_suresi	time		Egzersiz çalışmalarının süre bilgisi
	resim_url	text		Egzersiz görüntüsünün url bilgisi.
	egzersiz_tip_id	int	11	İlgili olaya referans.
	egzersiz_sirasi	smallint	6	Yapılacak egzersiz çalışmasının sıralama bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
egzersiz_tipleri	egzersiz_tip_id	int	10	İlgili olaya referans.
	egzersiz_tip	varchar	50	Egzersiz hareketleri tür bilgisi.
kullanicilar	kullanici_id	int	11	İlgili olaya referans.
	email	varchar	100	Kullanıcı email bilgisi.
	sifre	varchar	32	Kullanıcı şifre bilgisi.
	google_id	varchar	100	Kullanıcı google hesap bilgisi.
	facebook_id	varchar	100	Kullanıcı facebook hesap bilgisi.
	kullanici_ad	varchar	50	Kullanıcı ad bilgisi.
	kullanici_soyad	varchar	50	Kullanıcı soyad bilgisi.
	dogum_tarihi	date		Kullanıcı doğum tarihi bilgisi.
	kullanici_kilo	float		Kullanıcı kilo bilgisi.
	cinsiyet_id	tinyint	4	İlgili olaya referans.
	kullanici_boy	float		Kullanıcı boy bilgisi.
	saglik_bilgileri	text		Kullanıcının sağlık bilgilerinin detaylı açıklama bilgisi.
	is_vejeteryan	tinyint	1	Kullanıcın vejeteryanlık tercih bilgisi.
	is_pro	tinyint	1	Kullanıcın pro üyelik bilgisi.
	dil_id	int	11	Kullanıcının tercih ettiği konuşma dili bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
odemeler	odeme_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	fiyat	varchar	10	Kullanıcının ödemesi gereken tutar bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma durumu.
	createdAt	timestamp		Tablo oluşturulma tarihi.
ogunler	ogun_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	besin_id	int	11	İlgili olaya referans.
	ogun_tip_id	int	11	İlgili olaya referans.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
ogun_tipleri	ogun_tip_id	int	2	İlgili olaya referans.
	ogun_tip	varchar	15	Öğün tip bilgisi. Kahvaltı, öğlen yemeği vs.
su_takip	su_takip_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	su_miktar	int	11	Kullanıcının içmiş olduğu su miktarı bilgisi.
	durum	int	11	Tablo durumunun aktif veya pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
tarifler	tarif_id	int	11	İlgili olaya referans.
	kullanici_id	int	11	İlgili olaya referans.
	tarif_adi	varchar	50	Tarif ad bilgisi.
	tarif_aciklama	text		Tarif hakkında detaylı açıklama.
	resim_url	text		Tarife ait görüntünün url bilgisi.
	kalori	float		Tarifin toplam kalori bilgisi.
	tarif_tip_id	int	11	İlgili olaya referans.
	is_vejeteryan	tinyint	1	Tarif vejeteryanlık bilgisi.
	durum	int	11	Tablo durumunun aktif pasif olma bilgisi.
	createdAt	timestamp		Tablo oluşturulma tarihi.
tarif_tipleri	tarif_tip_id	int	10	İlgili olaya referans.
	tarif	varchar	20	Tarif tip ad bilgisi.(Yemek tarifleri vb.)

Şekil 3.13: Veri Sözlüğü.

3.2.7 İşlevlerin Sıradüzeni

Kullanıcı Google, Facebook veya mail hesabı ile sisteme giriş yapar. Sınırlı erişim ile devam eden kullanıcı hazır diyet listeleri, tarifler ve egzersizlerin bir kısmını görüntüleyebilir. Sınırsız erişim için Premium üyelik satın alır. Admin Premium üyeliği onaylar ve ücret yönetimini gerçekleştirir. Premium kullanıcı diyetisyen iletişime geçebilir ve kişiye özel diyet planı talebinde bulunabilir. Diyetisyen kullanıcı ile iletişime geçerek talep edilen diyet planını oluşturur.



Şekil 3.14: Bazı İşleyiş Diyagramları.

3.2.8 Başarım Gerekleri

Mevcut sistemler incelenmiştir ve mevcut sistemin eksiklerinden yola çıkılarak, sistemin başarımı için:

- Sistemin sonuç üretim doğrulukları
- Tepki sürelerinin en aza indirilmesi
- Mali külfetin azaltılması
- Kullanım kolaylığı
- Anlaşılabilirlik
- Tarafsızlık

temel gereklilikler olarak tespit edilmiştir.

3.3 Arayüz Gereksinimleri

3.3.1 Kullanıcı Arayüzü

Kullanıcı arayüzünde kullanıcı için sade ve basitleştirilmiş bir arayüz tasarlanması gerekmektedir. Kullanıcının mobil becerisinin en alt düzeyde olduğu düşünülmüştür.

3.3.2 Donanım Arayüzü

Kullanım için mobil cihaz yeterlidir. İnternet bağlantısı muhakkak gereklidir.

3.3.3 Yazılım Arayüzü

Kullanıcı sisteme girdiğinde erişim türünü seçer. Kullanıcının tercih ettiği sınırlı ve sınırsız erişime göre etkinliklerden kısıtlı ve kısıtsız olarak yararlanır.

3.4 Belgeleme Gerekleri

3.4.1 Geliştirme Sürecinin Belgelenmesi

Geliştirme süreci belirtilmiş olan kriterler doğrultusunda gerçekleştirilmiştir.

4. SİSTEM TASARIMI

4.1 Genel Tasarım Bilgileri

4.1.1 Genel Sistem Tanımı

Tablo 4.1: Genel Sistem Tanımı.

Gereksinimler

Kullanıcıların ihtiyaçları doğrultusunda piyasadaki uygulamalardaki eksiklikler üzerinde araştırma yapımı ve kullanıcı ihtiyacını karşılayacak gereksinimleri saptamak asıl amaçlardan biridir.

İşlevsel Belirtimler

Kullanıcıların kayıt ekranında girdikleri bilgiler ile ideal kilo hesabına uygun diyet listeleri, yemek tarifleri, su bildirimleri alır.

Adminler, bildirileri düzenleme/ekleme/güncelleme işlemlerini ayarlar. Kullanıcıların Premium üyeliğini onaylar.

Diyetisyenler, kullanıcılara özel diyet listeleri hazırlayabilir.

Tasarım

Tasarım aşamasında izlenecek adımlar sırayla şu şekildedir;

- Süreç Tasarımı
- Arayüz Tasarımı
- Yapısal Tasarım
- Veri Tasarımı

4.1.2 Sistem Mimarisi

Şekil 4.2: Veri Tabanı Sistemi.

4.1.3 Dış Arabirimler

4.1.3.1 Kullanıcı Arabirimleri

Kullanıcı arabirimlerine erişim, kullanıcıların ilk ekranda girdikleri kişisel bilgiler ile sağlanır. Kullanıcıların sistem içerisinde kendilerine ait bilgileri tutulmakta olup düzenlenebilen ve saklanabilenen profil yapısı bulunmaktadır.

4.1.3.2 Veri Arabirimleri

Veri arabirimlerinde sistem NodeJS ile çalışacağından veri tabanından kayıt yazıp okurken arada JSON arabirimleri olacaktır.

4.1.4 Testler

Genel hatlarıyla testlerimiz iki aşamada gerçekleştirilecek.

Alfa Aşaması: Sistemin geliştirildiği yerde kullanıcılar tarafından test işlemi gerçekleştirilecek.

Beta Aşaması: Kullanıcı, geliştirilen sistemi kendi ortamında test edecek.

4.1.4.1 Entegrasyon ve Test Gereksinimleri

Var olan sistemi güncel tutabilmek için sürekli olarak istekler doğrultusunda güncellemeler eklenmelidir. Bunun için yeni eklentilerde hata çıkmaması ve sistemde arızalara sebep olmaması için entegrasyon testleri gerekmektedir.

4.1.5 Performans

Sistemin her şart altında kesintisiz çalışarak kullanıcıya hizmet vermesi gerekmektedir. Bu nedenle gerçekleştirilen sistemin performansını etkileyen faktörlerin test verileri değerlendirilecektir.

- Sistemin Tasarıma Uygunluk Performansı: Tasarımı yapılan sistemin uygunluk ve işleyiş performansı değerlendirilecek.
- Veri Yapısının Sistemle Performansı: Veri yapısının sistemle uygunluğunu ve çalışma zamanındaki uyumluluk düzeyindeki performansı değerlendirilecek.

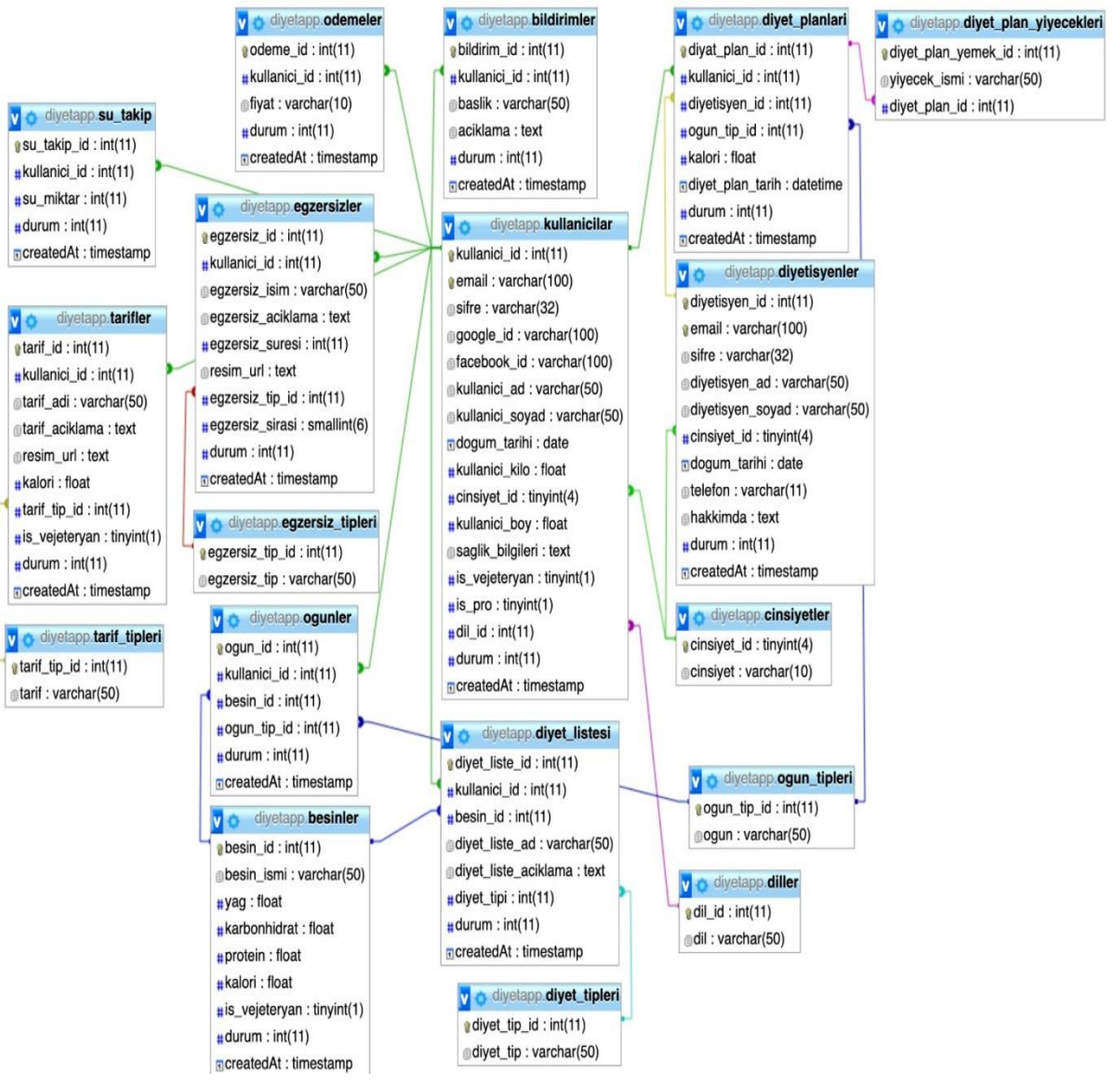
4.2 Veri Tasarımı

4.2.1 Tablo Tanımları

Sistem 18 tablodan oluşmaktadır. Her tablonun detaylı açıklaması ekte yer almaktadır.

4.2.2 Tablo-İlişki Şemaları

İlişkisel veri tabanı veri modelinde veriler tablolar üzerinden kurulan ilişkiye dayanmaktadır.



Şekil 4.3: Tablo-İlişki Şeması.

4.3 Süreç Tasarımı

Sistemin süreç tasarımında aşağıda belirtilen adımlar sırasıyla gerçekleştirilmektedir:

- Tasarımın Genel Özelliklerini Belirleme
- Taslak Tasarım Önerisi Geliştirme
- Tasarım Önerisine Yönelik Araştırma
- Tasarım Önerisini Geliştirme
- Yapım
- Değerlendirme ve Test Etme
- Değişiklik Önerme
- Pazarlanabilir Hale Getirme
- Sorunu Araştırma, Tanımlama ve Çözümünü Tartışma

5. SİSTEM GERÇEKLEŞTİRİMİ

5.1 Yazılım Geliştirme Ortamları

Tasarım sonunda üretilen fiziksel modeli bir bilgisayar ortamında çalıştırmak için bir yazılım geliştirme ortamı kullanılmalıdır:

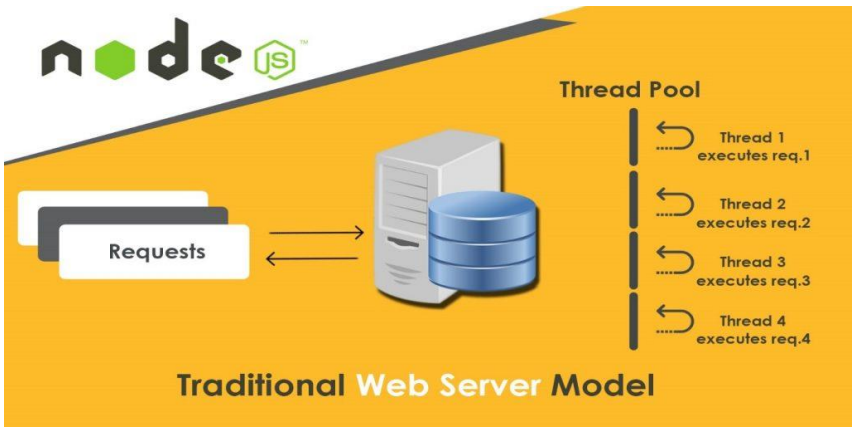
1. Programlama Dili
2. Veri Tabanı Yönetim Sistemi
3. Hazır Program Kitapçıkları
4. CASE aracı

belirlenir ve yazılım geliştirme ortamı hazırlanır.

5.1.1 Programlama Dilleri

Programlama dili olarak Facebook tarafından geliştirilmiş olan React Native ve React JS frameworkleri kullanılmıştır. Bu framework Javascript'den gücünü aldığı için proje geliştirmesi oldukça basit ve dinamik olacaktır. React Native ve React JS tasarım konusunda da oldukça basit ve kolay kodlama sunmaktadır. Veri tabanı işlemlerinde ise NodeJS ve MySQL kullanılmıştır. NodeJS sayesinde veri tabanını oldukça hızlı ve basit bir şekilde yönetmekteyiz.

5.1.2 Veri Tabanı Yönetim Sistemleri

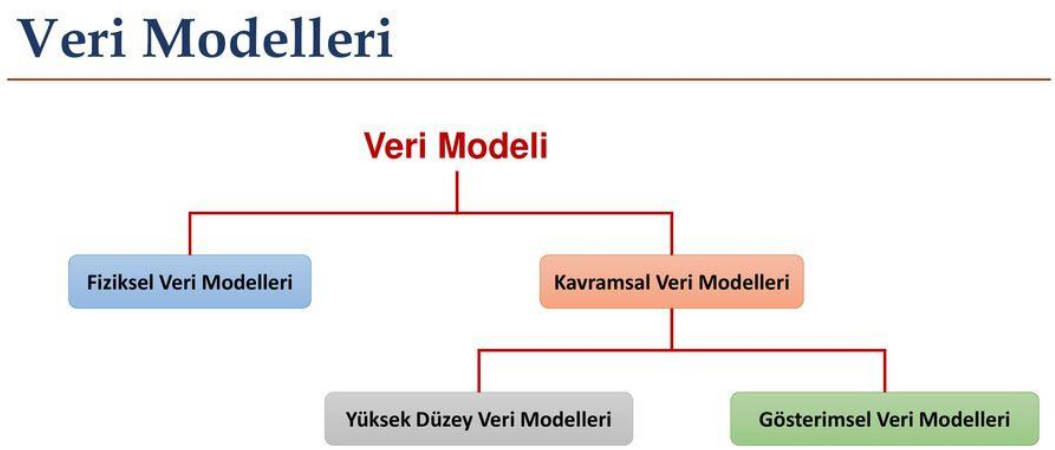


Şekil 5.1: Veri Tabanı Sistemi.

5.1.2.1 VTYS Kullanımının Ek Yararları

- Genişleme potansiyeli, esneklik ve uygulama geliştirme hızını arttırması.
- Güncel bilgilerin tüm kullanıcılara aynı zamanda ulaşması.
- Ortak verilerin tekrarının önlenmesi, verilerin merkezi denetiminin ve tutarlılığının sağlanması.
- Aynı veriler farklı bilgisayarlarda tekrar tekrar depolanmaz; veri tekrarı azaltılır veya yok edilir.
- Aynı verilerin farklı yerlerde kopyalarına sahip olmak bakım zorlukları yaratır. Bir yerde güncellenen adres bilgileri başka bir yerde değişmeden kalabilir ve bu da veri tutarsızlıklarına yol açabilir.
- DBMS'nin veri paylaşmak için kullanılmadığı durumlarda, kullanıcı aynı verilere aynı anda erişemez ve DBMS'de saniyede aynı veri tabanlarına yüz binlerce erişim sağlanabilir.
- Bir kayıt veri bütünlüğü tablosundan kaldırılırsa, o kayıtın bilgilerinin bulunduğu diğer tablolardan da kaldırılır.
- Veri hasarını önlemek için çok katlı mekanizmalara sahiptir.
- Veritabanına girmek için kullanıcı adı ve parola ile giriş yapmanın yanı sıra, insanlar yalnızca kendileri için geçerli olan tabloları veya tablodaki belirli sütunları görebilirler.

5.1.2.2 Veri Modelleri



Şekil 5.2: Veri Modeli.

Fiziksel veri modelinde, verilerin MySQL'deki tablolarda saklanacağını ve birbirleriyle ilişkili olduğunu söyleyebiliriz. Kavramsal veri modeli ise iki ana başlık altında incelenilmektedir.

5.1.2.3 Veri Tabanı Dilleri ve Arabirimleri

Neredeyse tüm sistemlerde uygun görülen, normal boyutlardaki verilerle kolayca başa çıkabilen MySQL kullanılmıştır. MySQL serbestçe kullanılabilen bir veritabanı sistemidir. Ancak, gelişmiş özelliklerinin kullanılabileceği ücretli bazı sürümleri de vardır. Diğer veritabanı yazılımlarıyla karşılaştırıldığında kullanımı daha kolaydır. Herhangi bir programlama dili ile kullanılabilmektedir. MySQL, meta verileri tanımlamak ve yönetmek için Linux, Windows, Unix gibi birden fazla platformda çalışabilir. Yerel sisteme ve hatta sunucuya kurulabilmektedir. Esnek, ölçeklenebilir, hızlı ve güvenilir bir çözümdür.

5.1.2.4 Veri Tabanı Sistem Ortamı

Tüm verileri yükleme, yedekleme, performans değerlendirme, sıralama, veri sıkıştırma gibi tüm işlevleri gerçekleştirmek için phpMyAdmin kullanılmıştır.

5.1.2.5 VTYS'nin Sınıflandırılması

En sık kullanılan veri modelleri ilişkisel modeller, ağ modelleri, hiyerarşik modeller, nesneye yönelik modeller ve kavramsal modellerdir. Projede ilişkisel veri modeli kullanılmıştır.

5.1.2.6 CASE Araç ve Ortamları

Case aracı olarak online web sitesi olan Creatly programı kullanılmıştır.

5.2 Kodlama Stili

Kod oldukça okunaklı ve temizdir. Süslü parantezler yeni satırda değil aynı satırda yazılır. Bir satırdaki karakter uzunluğunun 100-120 karakteri geçmemesine özen gösterilmiştir. Çünkü yatay kodu takip etmesi zordur. Javascriptte noktalı virgül kullanımı zorunlu değildir. Ancak her cümlenin sonuna noktalı virgül koyulmaya özen gösterilmiştir. Çok fazla iç içe kod yazılmamaya çalışılmıştır. Kod stili için otomatik düzenleyiciler kullanılmıştır. Bu denetleyici son zamanlarda popüler olmaya başlayan ESLİNT'dir.

5.2.1 Açıklama Satırları

Açıklama satırları proje ekiplerinin birbirleri ile senkronize olmasını sağlar. Bu projede de olabildiğince fazla açıklama satırları kullanmaya çalışılacaktır. Her karmaşık satırın sonuna bir açıklama satırı eklenecektir. Her günün sonunda tarih, kodun tutulduğu yere yerleştirilmektedir.

5.2.2 Kod Biçimlemesi

Kod biçiminde, temel koddaki doğal dizinleri kullanılmıştır ve iç içe bir şekilde hiyerarşi oluşturulmuştur.

5.2.3 Anlamlı İsimlendirme

Değişkenler, program yürütme sırasında oluşturulan ve veri depolamak için kullanılan öğelerdir. Değişkenler oluşturulurken, isimleri ve veri türleri belirlenir. Bu isim, bir değişkene değer atamak istendiğinde veya bir değişken tarafından depolanan değere ulaşılmak istendiğinde kullanılır. JavaScript büyük / küçük harfe duyarlı bir dildir. Değişkenleri adlandırırken Türkçe karakterler (ı, İ, ğ, Ğ, ü, Ü, ş, Ş, ö, Ö, ç, Ç), boşluklar ve özel karakterler (.,; / vb.) kullanılmamalıdır. Benzer şekilde, programlama dillerinde farklı anlamları olan sözcükler değişken adları olarak seçilmemelidir. (Int, not, char, vb.) Değişken adları sayılarla veya rakamlarla başlayamaz. Değişkenler note1 olarak adlandırılabilir, ancak 1note olarak adlandırılamaz.

5.3 Olağandışı Durum Çözümleme

Olağan dışı durum, program işleminin geçersiz veya yanlış veri üretimi ve beklenmedik diğer nedenlerle sona erdiği bir durum olarak tanımlanır.

5.3.1 Olağandışı Durum Tanımları

Try-catch bloğunun anormal gelişme durumlarında rol oynayabileceği ve programı kesintisiz yürütmeye devam edebileceği şekilde tasarlandı.

5.3.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

Tüm özel durumlarda, programı bozmadan bir hata mesajı verecek şekilde tasarlandı.

5.4 Kod Gözden Geçirme

Önceki sürümü gözden geçirmeden okunabilir bir program yazmak imkansızdır. İçerik onayı olmadan hiçbir makale basılamaz; hiçbir program gözden geçirilmeden çalıştırılamaz. Kod incelemesi ve program testi arasında ayrım yapmak gerekir. Program testinin amacı, program yürütülürken oluşabilecek hataları yakalamaktır. Kod incelemesi, program kaynak kodunun gözden geçirilmesidir. Kod incelemesinde, program hatalarının % 3-5'i yakalanabilir. Programı yazan kişi, yazdığı programın "kod inceleme" sürecine gireceğini biliyorsa, daha verimli, daha az hata ve daha okunabilir bir program alınmaktadır.

5.4.1 Gözden Geçirme Sürecinin Düzenlenmesi

Denetim sürecinin temel özellikleri şunlardır:

- Hataları bulup düzeltmek yerine keşfetmeyi amaçlar. Mümkün olduğunca küçük bir grup tarafından yapılmalıdır. Birden fazla kişiye ihtiyaç duyulursa, bakım yapan bir ekipten yararlanmak faydalıdır.
- Kalite çalışmanın bir parçası olarak düşünülmeli, sonuçlar sabit ve kesin bir şekilde saklanmalıdır. Burada cevaplanacak temel soru, programın yazılı olarak çalışıp çalışmadığını belirlemektir.

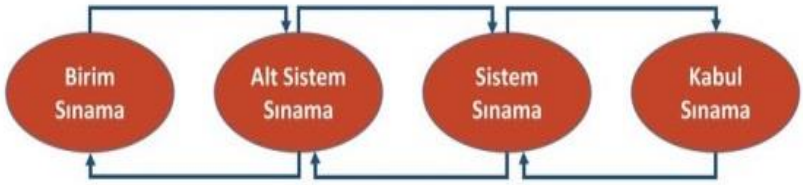
6. DOĞRULAMA VE GEÇERLEME

Yazılım geliştirme karmaşık bir süreç olduğundan hataların ortaya çıkması kaçınılmazdır.

- Yazılım yaşam döngüsünün her aşamasında hatalara karşı sınanması,
- Gereksinimler arasındaki tutarsızlıkların giderilmesi,
- Çözümleme şeması ile uygulama arasındaki uyumsuzlukların giderilmesi,
- Tasarım hatalarının düzeltilmesi,
- Çalışma anı hatalarının giderilmesi,
- Yaşam döngüsü ilerledikçe, hataların düzeltilmesi,

İyi bir sınama yaklaşım hatalarının erken belirlenmesine katkıda bulunacak ve yazılım kalitesini arttıracaktır.

6.1 Sınama Kavramları



Şekil 6.1: Sınama Kavramları

Birim sınama: Bağlı oldukları diğer sistem unsurlarından tümüyle soyutlanmış olarak birimlerin doğru çalışmalarının belirlenmesi amacıyla yapılır.

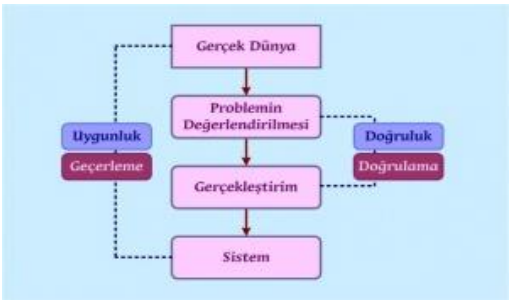
Alt sistem sınama: Alt sistemler modüllerin bütünleşmeleri ile ortaya çıkar. Yine bağımsız olarak sınamaları yapılmalıdır.

Sistem sınama: Üst düzeyde, bileşenlerin sistem ile olan etkileşiminde olan hatalar aranmaktadır. Ayrıca belirtilen ihtiyaçların doğru yorumlandıkları da sınanmalıdır.

Kabul sınaması: Çalıştırılmadan önce sistemin son sınanmasıdır. Artık yapay veriler yerine gerçek veriler kullanılır. Bu sınama türü alfa sınaması veya beta sınaması olarak da bilinir.

6.2 Doğrulama ve Geçerleme Yaşam Döngüsü

Yaşam döngüsü doğrulama ve geçerleme işlemleri yazılım üretim yaşam döngüsünün tüm süreçlerinde ve bu süreçlere koşut olarak sürer. Gerçekleştirim aşamasına kadar olan süreçlerde doğrulama ve geçerleme işlemlerinin planlanması yapılır. Planlama genel olarak, birim, alt sistem, bütünleştirme, sistem ve kabul sınamalarının tasarımlarını içerir. Gerçekleştirim aşamasının sonunda ise söz konusu planlar uygulanır. Uygulama sonucu elde edilen bulgular, Yazılım Doğrulama ve Geçerleme raporları biçiminde sürekli olarak raporlanır. Bu bilgiler, değişiklik denetim sistemi ve sorun yönetim sistemlerinde girdi olarak kullanılır. Değişik Denetim sistemi, sınama süresince elde edilen bulguların izlenmesi amacıyla oluşturulan bir sistemdir. Bu sistemde, sınama sonucu elde edilen bulgular ve bunlara karşı sorun yönetimi tarafından alınan önlemler izlenir.

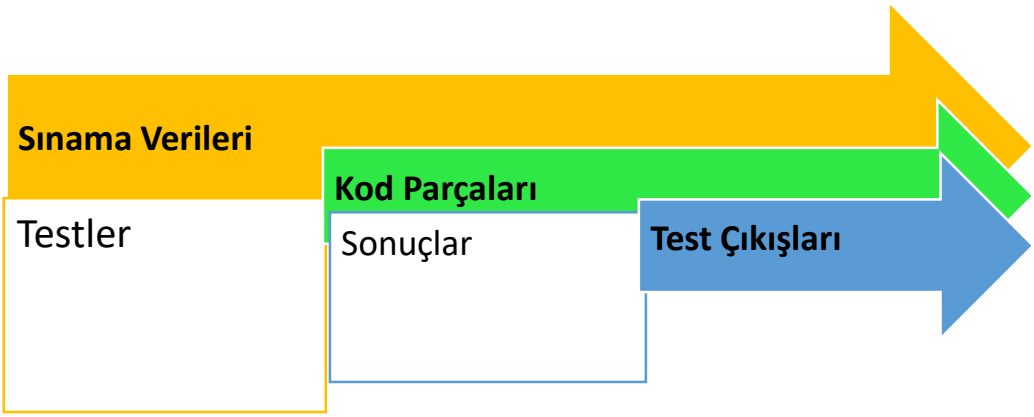


Şekil 6.2: Doğrulama ve Geçerleme Yaşam Döngüsü.

6.3 Sınama Yöntemleri

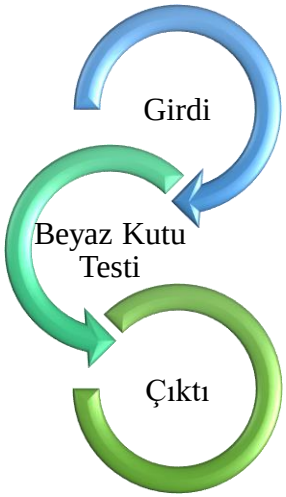
Sınama işlemi, geliştirmeyi izleyen bir düzeltme görevi olmak ile sınırlı değildir. Bir "sonra" olmaktan çok, geliştirme öncesinde planlanan ve tasarımı yapılması gereken bir çaba türüdür. Her mühendislik ürünü, iki yoldan biri ile sınanır: sistemin tümüne yönelik işlevlerin doğru yürütüldüğünün (kara kutu – black box) veya iç işlemlerin belirtimlere uygun olarak yürütüldüğünün bileşenler tarafından sınanması

(beyaz kutu – white box). Kara kutu sınamasında sistemin, iç yapısı bilinmeksizin gelişigüzel girdiler verilerek sınama yapılır. Sonraki sayfalarda beyaz kutu sınamasına ilişkin bilgiler verilmektedir.



Şekil 6.3: Yazılımın Sınanması

6.3.1 Beyaz Kutu Sınaması



Şekil 6.4: Beyaz Kutu Testi

Beyaz kutu sınaması tasarlanırken, birimin süreç belirtiminden yararlanılır. Yapılacak denetimler arasında:

- Bütün bağımsız yolların en azından bir kere sınanması,
- Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması,
- Bütün döngülerin sınır değerlerinin sınanması
- İç veri yapılarının denenmesi bulunur. Sınamaları yürütürken sınırlı çabamızı yerinde kullanmamız gerekir.

Bunun için hataların bazı özelliklerinin bilinmesinde yarar vardır:

- Bir program kesiminin uygulamada çalıştırılma olasılığı az ise o kesimde hata olması ve bu hatanın önemli olması olasılığı fazladır.

- Çoğu zaman, kullanılma olasılığı çok az olarak kestirilen program yolları, aslında çok sıkça çalıştırılıyor olacaktır.
- Yazılım hataları rasgele olarak dağılır. Bunlardan bazılarını derleyiciler bulur, bazılarıda bulunmadan kalır.

6.3.2 Temel Yollar Sınaması

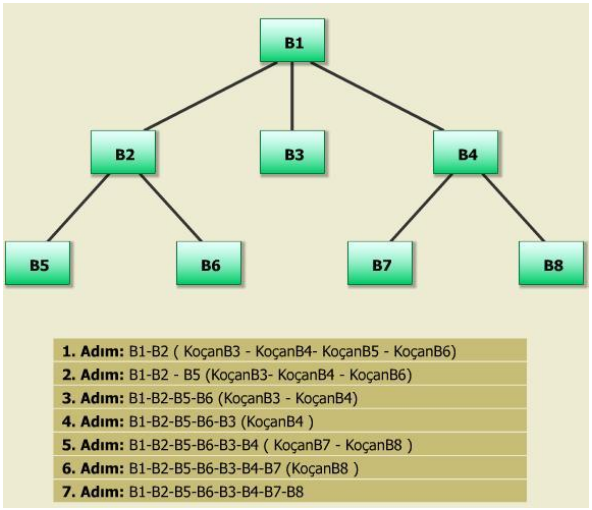
Temel yollar testi, işlevsel tasarımın mantıksal karmaşıklığını ölçmek ve bu ölçüye göre uygulama yolları için bir temel grup oluşturmak esasına dayanmaktadır. Bu grubu denemek için bir test programı düzenlemektedir. Test programları, test sırasında programdaki her deyimi en az bir kez uygulayarak denemektedir.

6.4 Sınama ve Bütünleştirme Stratejileri

Genellikle sınama stratejisi, bütünleştirme stratejisi ile birlikte değerlendirilebilir. Ancak bazı sınama stratejileri bütünleştirme dışındaki tasaları hedefleyebilir. Örneğin, yukarıdan aşağı ve aşağıdan yukarı stratejileri bütünleştirme yöntemine bağlıdır. Ancak işlem yolu ve gerilim sınamaları, sistemin olaylar karşısında değişik işlem sıralandırılmaları sonucunda ulaşacağı sonuçların doğruluğunu ve normal şartların üstünde zorlandığında dayanıklılık sınırını ortaya çıkarır.

6.4.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

Yukarıdan aşağıya bütünleştirmede önce sistemin üst düzeylerinin sınanması ve sonra aşağıya doğru olan düzeylere ilgili modülleri takılarak sınanması söz konusudur. En üst noktadaki bileşen sınandıktan sonra alt düzeye geçilmelidir. Alt bileşenler henüz hazırlanmamışlardır. Bu sebeple koçanlar kullanılır. Koçan, bir alt bileşenin, üst bileşen ile arayüzünü temin eden fakat işlevsel olarak hiçbir şey yapmayan çerçeve programlardır.

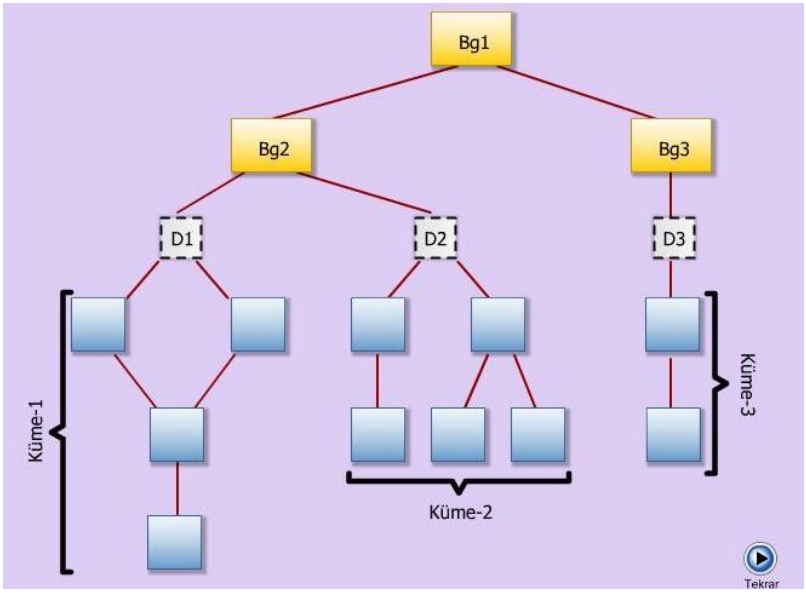


Şekil 6.5: Yukarıdan aşağıya bütünleştirme örneği

6.4.2 Aşağıdan Yukarı Sınama ve Bütünleştirme

Aşağıdan yukarı bütünleştirmede ise, önceki yöntemin tersine uygulama yapılır. Önce en alt düzeydeki işçi birimler sınanır ve bir üstteki birimle sınama edilmesi gerektiğinde bu üst bileşen, bir 'sürücü' ile temsil edilir. Yine amaç, çalışmasa bile arayüz oluşturacak ve alt bileşenin sınanmasını sağlayacak bir birim edinmektir. Bu kez kodlama, bütünleştirme ve sınama aşağı düzeylerden yukarı düzeylere doğru gelişir ve yukarı düzeylerde önce sürücü olarak yazılan birimler sonra gerçekleriyle yer değiştirerek o

düzeyin birimleri/alt sistemleri olurlar. Bütünleştirme yukarı doğru yapıldıkça daha az sürücü gereği duyulur.



Şekil 6.6: Aşağıdan yukarıya bütünleştirme

Uygulamada, hem aşağıdan yukarıya, hem de yukarıdan aşağıya sınama stratejileri kullanılacaktır.

6.5 Sınama Planlaması

Sınama işlemi çok kapsamlıdır. Bir plan güdümünde gerçekleştirilmelidir. Böyle bir planın temel bileşenleri önceki sayfalarda belirtilmiştir. Yazılım yaşam döngüsünün süreçlerine koşut olarak, farklı ayrıntı düzeylerinde birden fazla sınama planı hazırlanır.

Sınama planları; Birim (Modül) Sınama Planı, Alt Sistem Sınama Planları, Bütünleştirme Sınama Planları, Sistem Sınama Planları biçimindedir. Her sınama planı, sınama etkinliklerinin sınırlarını, yaklaşımını, kaynaklarını ve zamanlamasını tanımlar. Plan neyin sınanacağını, neyin sınanmayacağını, sorumlu kişileri ve riskleri göstermektedir. Sınama planları, sınama belirtimlerini içerir.

6.6 Sınama Belirtimleri

Sınama belirtimleri, bir sınama işleminin nasıl yapılacağına ilişkin ayrıntıları içerir. Bu ayrıntılar temel olarak:

- Sınanan program modülü ya da modüllerin adları,
- Sınama türü, stratejisi (beyaz kutu, temel yollar vb.),
- Sınama verileri,
- Sınama senaryoları türündeki bilgileri içerir.

Sınamayı yapan, sınama tarihi bulunan hatalar ve açıklamaları türündeki bilgiler eklenerek sınama raporları oluşturulur.

6.7 Yaşam Döngüsü Boyunca Sınama Etkinlikleri

Sınama etkinlikleri, üretim aşamasının en başından başlayarak planlanır, ayrıntılandırılır ve raporlanır. Sınama etkinliklerinin yaşam döngüsü çekirdek adımlarındaki dağılımı gösterilmektedir. Yapılan işe ilişkin bir düzeyi arttıkça, sınama planları soyut durumdan somut sınamalara dönüşür

Hazırlanan sınama raporları, doğrulama ve geçerleme yaşam döngüsü işlemleri gereği "sorun yönetimi" ne iletilir. Bu bölümde hatalar kaydedilir ve bulunan hatalara karşı yapılacak işlemler planlanır. Sınama sırasında bulunan her bulgu ya da hata olarak belirtilen her durum gerçekte hata olmayabilir. Farklı sınayıcılardan biri, bir durumu hata olarak nitelerken diğeri aynı durumu doğru olarak değerlendirebilir. Bu nedenle sınama raporlarında hata olarak bildirilen her durum hemen düzeltilmek üzere ele alınmaz. Önce çözümlenir, kullanıcı çelişkileri giderilir ve gerçekten hata olduğuna karar verilirse düzeltilir. Söz konusu karar kullanıcı temsilcileri ile birlikte alınır. Sınama sırasında bulunan her hata için, değişiklik kontrol sistemine (DKS), "Yazılım Değişiklik İsteği" türünde bir kayıt girilir. Hatalar, DKS kayıtlarında aşağıdaki gibi gruplara ayrılabilir:

Şekil 6.7: Sınama Etkinlikleri

Şekil 6.7. Gerçekleştirme ve Doğrulama Zaman diyagramı

7. BAKIM

Sistemin tasarımı ve kodlaması bittikten sonra sınanan sistemin, kullanıcı alanına yüklenmesi ve kullanıma başlandıktan sonra ortaya çıkan yazılım kusurlarının giderilmesi için düzenli aralıklarla veya ihtiyaç sürelerinde bakıma sokulması gerekir.

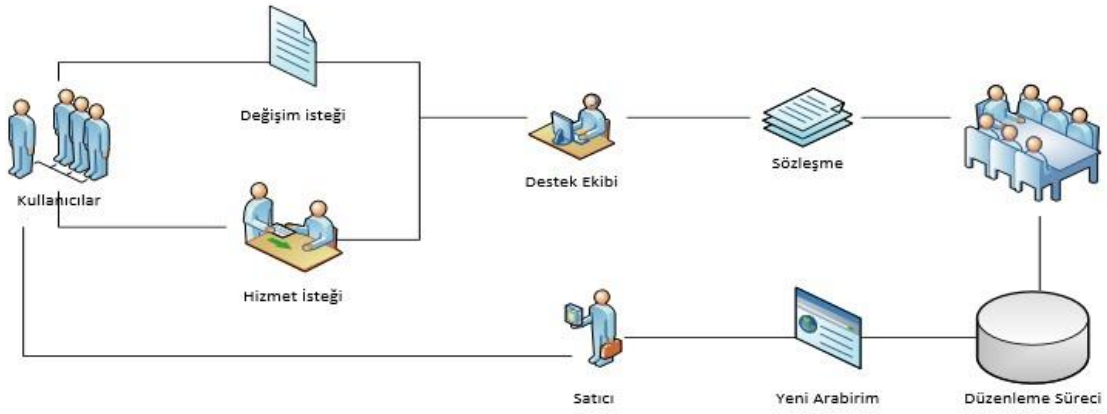
7.1 Kurulum

Kullanıcılara özel bir kurulum olması sebebiyle kullanıcıların erişebilmesi açısından App Store ve Play Store mobil mağazalarından telefonlara kurulum sağlanabilir.

7.2 Yerinde Destek Organizasyonu

Kurulum kullanıcıların kendi sahalarında olması sebebiyle oluşabilecek bir aksaklık durumunda mobil mağazalardan geliştirici ekip ile iletişime geçebilir. Geliştirici ekip bir aksaklık durumunda bu şekilde müdahale edebilir.

7.3 Yazılım Bakımı



Şekil 7.1: Bakım Süreci

Bakım, işletime alınan yazılımın sağlıklı olarak çalışması ve ayakta kalabilmesi için yapılması gereken çalışmalar bütünü olarak tanımlanır. Sistemlerde üç tür bakım gereksinimi bulunmaktadır. Sırayla bu bakımlar sistemimizde kullanılması amaçlanmaktadır.

Düzeltici Bakım

Bir yazılım %100 test edilemeyeceğinden zaman zaman hataların ortaya çıkan hataların düzeltilmesine düzeltici bakım denir. Sistemimizde bakım çalışmalarında bulunmaktadır.

Uyarlayıcı Bakım

Sürekli gelişen bir sistem yapısında bazı günler durağan olabilirken bazı günler fazla veri girişi olabilir. Sistem birden fazla kullanıcı gereksinimine ihtiyaç duyabilir. Bu sebeple yeni eklentilere uyarlayıcı bakım uygulanmaktadır.

En iyileyici Bakım

Uygulama yazılımlarında hız performansı veya çalışma performanısının iyileştirilmesi amacıyla yapılacak bakım çalışmalarıdır.

7.3.1 Bakım Süreç Modeli

Bakım süreci yazılım yaşam döngüsünün temel adımlarının tekrarı bu kısımda da geçmektedir. IEEE 1219 standartı tarafından önerilen süreç modelidir. Sistem bakıma sokulduğunda aşağıdaki süreçler sırasıyla uygulanmalıdır.

- Sorun Tanımlama Süreci
- Çözümleme Süreci
- Tasarım süreci
- Gerçekleştirim Süreci
- Sistem Sınama Süreci
- Kabul Sınama Süreci
- Kurulum Süreci