**Fonksiyonel Gereksinimler**

1. **Sisteme Giriş:** Kullanıcı/Müşterilerin sisteme gerekli bilgilerini girdikten sonra giriş yapabilmeli ve sistem özelliklerini kullanabilmeli.
2. **Kullanıcı/Müşteri Kaydı Oluşturma:** Müşterilerin sistem üzerinden araç kiralama işlemlerini yapabilmesi için sisteme kayıt oluşturması sağlanmalı.
3. **Araçların Listelenmesi:** Sistemdeki farklı sınıf ve modeldeki araçlar listelenmeli. Listelenen sayfada araç hakkında detaylı bilgi verilmeli.
4. **Rezervasyon İşlemleri:** Müşteriler gerekli doğrulama işlemlerini yaptıktan sonra seçtikleri araca ait rezervasyon yapabilmeli.
5. **Doğrudan Kiralama İşlemi:** Müşteri, aracı hemen kiralama durumunda kredi kartı ile ödeme işlemini yaparak aracı doğrudan kiralayabilmeli.
6. **Doğrulama İşlemi:** Müşterilerin kimliği, telefonlarına gelen SMS ile doğrulanabilmeli.
7. **Müşteriye Mail Gönderimi:** Müşteriye rezervasyon yaptıktan sonra veya diğer gerekli durumlarda otomatik mail gönderimi yapılabilmeli. (Kiralama işleminden sonra mail üzerinden e-fatura gönderimi)
8. **Yetkilendirme:** Yetki sahibi çalışanlar, sistem üzerinden gerekli yetkilendirmeleri yapabilmeli. (Yeni araç Ekleme, Güncelleme, Silme yetkileri)

**Teknik Gereksinimler**

* **Multi Platform Desteği:**
* Kullanılacak sisteme hem mobil uygulama üzerinden hem de web sitesi üzerinden erişilebilmeli.
* Mobil tarafta Android ve iOS işletim sistemlerinin ikisinde de sorunsuz kullanılabilmeli.
* Web ve Mobil arasındaki uyumluluk sağlanmalı.
* **Performans:** Sistem yoğun kullanım olduğu durumlarda bile sorunsuz şekilde çalışacak bir altyapıya sahip olmalı.
* **Güvenlik:**
* Müşteri bilgilerinin güvende kalması ve dışarıdan erişilememesi için gerekli ve güncel protokollere sahip bir güvenlik duvarı olmalı.
* Yapılacak herhangi bir siber saldırıda güvenlik sağlanabilmeli ve güvenlik açıkları belli aralıklarla test edilip sistem güvenliği her daim sağlanmalı.
* **Ödeme İşlemi:**
* Kredi kartı ile ödeme yapılması durumunda güvenliği sağlamak için gerekli ödeme teknolojileri kullanılmalı.
* Kullanılacak teknoloji *PCI DSS* standartlarına uymalı.
* **Veri tabanı:** Araç, müşteri ve log bilgilerini depolayabilmek için performanslı ve veri bütünlüğü olan bir veri tabanı kullanılmalı.
* **Loglama İşlemleri:** Sistemde yapılan her hareketi sonradan incelenebilecek şekilde kayıt altına alınabilmeli.
* **Server/Sunucu:**
* Server’ın performanslı çalışabilmesi için yeterli donanım ve altyapı kullanılmalı.
* Düzenli aralıklarla yedeklemeler alınmalı.
* Server 7/24 sorunsuz çalışabilecek kapasitede olmalı.
* Server’da oluşacak sorunlar sebebiyle sistemde. Kesintiler yaşanacak olursa bu durum yetkili kişilere uyarı şeklinde iletilebilmeli.
* **API Kullanımı:**
* Kullanılacak olan API’nin detaylı şekilde dokümantasyonu yapılmalı.
* API’nin güvenliği sağlanmalı.
* API’nin performanslı çalışması için gerekli optimizasyonlar yapılmalı.
* **Uygun Yazılım Mimari Stil Seçimi:**
* Projeye en uygun mimariyi seçip mimariye uygun altyapı sağlanmalı.

**Kalite Nitelikleri**

**Performans**

1. **Veri Tabanıyla İlgili Performans Senaryosu:**

* Büyükverilere sahip Veri tabanını karmaşık sorgular ile zorlayıp performansının yeterliliğini test edeceğiz.
* Eğer karmaşık sorgulara rağmen hızlı şekilde sonuçları geri döndürebiliyorsa Veri tabanımız performanslı şekilde çalışıyor diyebiliriz.

**Senaryo işleyişi:**

* Müşteri sisteme giriş yapıp herhangi bir aracı filtreleyerek araçları listeler.
* Bu filtreleme sonucunda araçlar sorunsuz ve hızlı şekilde müşterinin karşısına çıkmalıdır.

1. **Sistem Yoğunluğuyla İlgili Performans Senaryosu:** Sistemde çok yoğun trafik olduğu bir durum olduğunu varsayalım.

* Sistemin yoğunluk durumlarında işlevini yerine getirip getirmediği test edilecektir.

**Senaryo İşleyişi:**

* Müşteri sisteme giriş yapıp herhangi bir işlem gerçekleştirsin.
* Sistemde binlerce müşterinin aynı anda işlem yaptığı durumu ele alıp analiz yapalım
* Server kapasitesi, sistemdeki kullanıcı sayısı ve işlem yanıt süreleri gibi değerlere bakılarak test edilmeli.
* Kullanıcı sayısı ve işlem yanıt süreleri en optimum duruma gelince test tamamlanır ve sistemin yoğun isteklere cevap verebildiği doğrulanır
* Aksi durumda ise sistemin yeterli donanım ve altyapıya sahip olmadığı anlamına gelir. Bu durumda gerekli güncellemeler ve optimizasyonlar yapılıp server daha performanslı hale getirilmelidir.

**Güvenlik**

1. **Verilerin Korunmasına İlişkin Senaryo:** Müşterilerin kişisel bilgilerine siber saldırı ile erişmeye çalışılan bir senaryoyu düşünelim.

* Bu senaryo sonucunda kişisel verilere erişimin güvenliğinin sağlandığını ve saldırgan işlemlerin tespit edildiği test edilecektir.

**Senaryonun İşleyişi:**

* Sisteme yetkisi olmadan sızmaya çalışan bir hacker var.
* Sisteme girmeye çalışan hacker sistem tarafından fark edilir ve gerekli güvenlik önlemleri devreye girer.
* Bu saldırı sistemdeki yetkili kişilere hemen uyarı şeklinde bildirilir.
* Hackerin işlemleri tespit edilir ve kayıt altına alınarak güvenlik önlemlerinin arttırılması amacıyla kullanılır. (Sistem zafiyetlerini en aza indirgemek için)
* Bu test sonucunda eğer hacker sisteme erişim sağlayamazsa sistemin güvenlik önlemleri sorunsuz şekilde çalışıyor demektir.

**Süreklilik**

1. **Sistemin 7/24 Çalışma Durum Senaryosu:** Sistemin herhangi bir hata veya sorun karşısında dahi kesintisiz çalışmaya devam edebilmesini test edeceğimiz senaryo.

* Sistemdeki olağanüstü durumlar sebebiyle kesinti olma durumlarında en kısa sürede ikincil sistem veya önlemler devreye girerek sistemin çalışmaya devam etmesi sağlanmalıdır.

**Senaryonun İşleyişi:**

* Sistem normal şekilde çalışırken herhangi bir kesinti olması durumunu ele alalım.
* Sistem kesintiye uğradığı zaman bu durum sistemdeki yetkili kişilere bildirilir ve müşterilere de giriş ekranında belirtilir.
* Sistem kesintisi durumunda devreye ikincil sistem girerek sistem kaldığı yerden çalışmaya devam eder.
* Bu kesinti süreci raporlanıp analiz edilir ve muhtemel kesintilerin önlenmesi için kullanılmak üzere kayıt altına alınır.

**Projede Kullanılacak Yazılım Mimari Stili**

Projemizde müşterinin isteği doğrultusunda hem mobil uygulama hem de web sitesi yapılacaktır. Bu istek karşısında kapsamlı olacak olan bu projede servisleri (Ödeme Servisi, Bildirim Servisi, Güvenlik Servisi…) birbirinden bağımsız şekilde oluşturup yönetmek daha verimli olacağından yazılım mimari stili olarak *Mikro Servis* Stilini kullanmak daha verimli olacaktır.

*Mikro Servis* Stili sayesinde oluşturduğumuz yönetilebilir küçük servislere hem Mobil Uygulama hem de Web Sitesi üzerinden erişim sağlayabileceğiz. Bu stili seçmemizdeki en büyük etken, farklı platformlarda hizmet veriyor olsak bile servisleri birbirinden bağımsız şekilde yönetebilecek ve güncellemeler yapabilecek olmamızdır. Servislerden herhangi birisinde hata meydana geldiğinde diğer servisler bundan etkilenmeyerek işlevlerini yerine getirmeye devam edecekler. Bu da proje geliştirme esnasında bize özellikle zaman konusunda büyük bir avantaj sağlamış olacak.

Mikro Servis Mimari Stilini kullanma gerekçelerimiz aşağıda maddeler halinde belirtilmiştir.

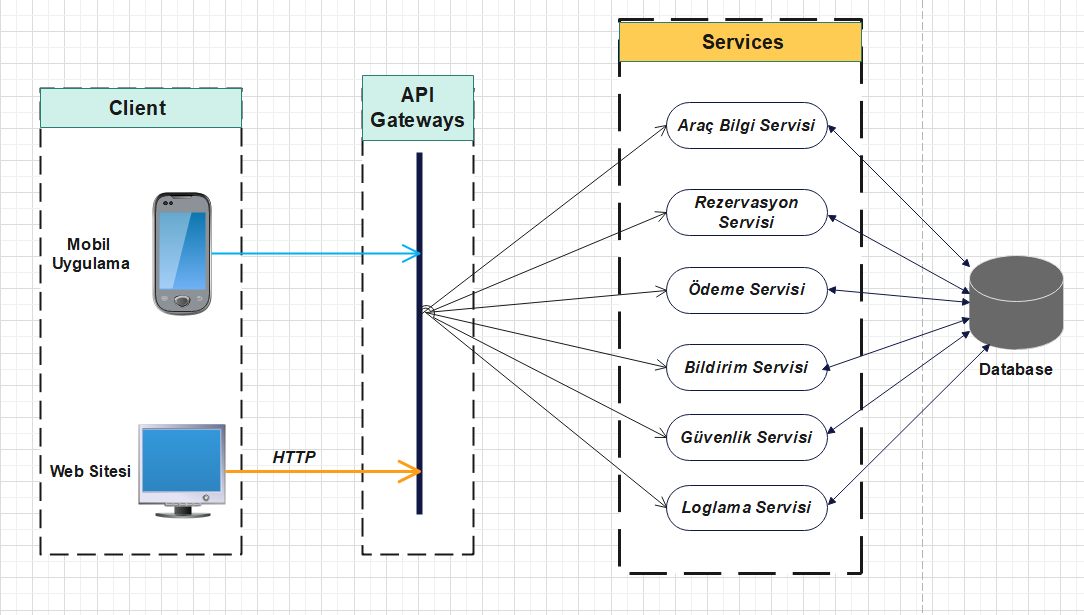
* Daha Yönetilebilir bir sisteme olanak sağlaması.
* Birbirinden bağımsız sitemlerin varlığı.
* Geliştirmeye açık olması.

**Projede Kullanılacak Servisler**

1. **Araç Bilgi Servisi:** Veri tabanındaki araçların listelenmesini sağlayan servis. Bu servis sayesinde araçlar çeşitli filtreleme işlemleri yapılarak listelenebilecek. Bu servis kullanıcının karşısına araçları ve araçlar hakkındaki bilgileri çıkaracaktır.
2. **Rezervasyon Servisi:** Bu servis kullanıcının *Araç Bilgi Servisi* üzerinden seçtiği aracın uygunluk durumunu kontrol eder ve uygun olması durumunda rezervasyon işlemini tamamlamak için *Ödeme Servisi ile iletişime geçer.*

Ayrıca bu servis sayesinde kullanıcılar rezervasyon iptali veya takip işlemlerini yapabilir.

1. **Ödeme Servisi:** Rezervasyon Servisinden gelen istek doğrultusunda ödeme işleminin güvenli şekilde yapılmasını sağlayan servistir.
2. **Bildirim Servisi:** Bu servis bildirim yapılması gereken durumlarda (Rezervasyon yapılması, Siber Saldırı…) gerekli kişilere SMS veya mail üzerinden bilgilendirme mesajı göndermekten sorumludur.
3. **Güvenlik Servisi:** Güvenlik servisi sistemin her türlü güvenliğinden sorumludur. Ödeme işleminden doğrulama işlemlerine kadar tüm işlemlerin güvenle yapılmasını sağlar.
4. **Loglama Servisi:** Sistemdeki tüm işlemleri kayıt altına alan servis. Bu servis sayesinde kayıtlar incelenerek analizler yapılır ve gerekli güncellemeler, yenilikler ve hatalar giderilir.



Görsel 1: Mikro Servis Mimarisi

**Kullanılacak Yazılım Geliştirme Modeli**

Projede müşterinin kendisi de dahil olacağından ve sürekli projenin durumu hakkında geri bildirim verecek olmasından dolayı *Agile (Çevik)* Yazılım Geliştirme Modelini kullanılacak

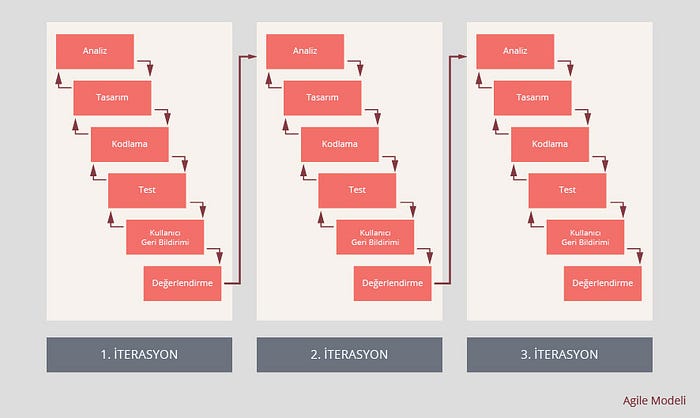
Projede *Agile* Modeline ait *SCRUM* Metodunu kullanma kararı verildi.

Agile-SCRUM Modelini kullanmamızdaki gerekçeler

* **Müşteri Geri Bildirimi:** Müşteri projenin her aşamasında proje hakkında geri bildirim vereceğinden müşterinin istediği sonuca ulaşmak daha kolay olacaktır.
* **Değişim İmkânı:** Olumsuz giden bir durum karşısında müşteri isteğine göre projeye yeniden yön verilebilir.
* **Ekip Ruhu/İletişim:** Sürekli yapılan toplantı ve değerlendirmeler sayesinde ekip içerisindeki iletişim üst düzeydedir. Dolayısıyla proje daha verimli şekilde yürütülebilir.
* **Yönetilebilirlik:** SCRUM sayesinde işler daha küçük parçalara ayrılıp daha yönetilebilir ve kontrol edilebilir yapıda olduğu için projede hız sağlıyor ve her sprint sonrasında iyileştirmeler yapılabiliyor.
* **Müşteri Memnuniyeti:** Müşteri en başından beri projeye dahil olduğu için istekleri karşılanmış ve memnun bir şekilde proje süreci tamamlanmış oluyor.

*Analiz, Tasarım, Kodlama, Test, Müşteri Geri Bildirimi, Değerlendirme* olmak üzere 5 aşamadan oluşuyor.

1. **Analiz:** Proje gereksinimleri belirlenir.
2. **Tasarım:** Proje mimarisinin ve arayüzün planlanma sürecidir.
3. **Kodlama:** Belirlenen gereksinimlere göre kodlama yapıldığı süreçtir.
4. **Test:** Kodlanmış projenin her türlü teste tabi tutulup doğru ve eksiksiz çalıştığı doğrulanır.
5. **Müşteri Geri Bildirimi:** Projede gelinen nokta müşteriye demo olarak sunulur ve müşteri bu demoya göre geri bildirim vererek projeye yön verir.
6. **Değerlendirme:** İlk 5 adımın içerdiği genel bir değerlendirme yapılır ve sonraki adım için yeni bir yol haritası çizilir.



Görsel 2: Agile-SCRUM Süreci

**Kullanılacak Tasarım Desenleri**

**Builder Deseni**

Builder desenini ***Araba*** sınıfında kullanmamızın sebebi araba sınıfından nesne oluştururken oluşabilecek karmaşanın önüne geçmektir. **Araba*Builder*** sınıfı sayesinde nesneye istediğimiz parametreleri atayabiliyoruz.

Builder Deseninin Kullanıldığı Sınıflar:

* Araba
* ArabaBuilder

**Factory Deseni**

Factory desenini kullanmamızdaki sebep nesne oluştururken daha soyut bir yapı kullanmak. Yeni eklenecek roller için tek bir sınıf kodunu değiştirmemiz yeterli olacağından sonraki süreçler için iş yükünü azaltmış olduk.

Factory Deseninin Kullanıldığı Sınıflar

* Kullanici (Interface)
* KullaniciFactory
* Musteri
* Yonetici

**Bridge Deseni**

Bridge desenini kullanmamızdaki amaç kiralanacak arabanın daha kolay şekilde işleme sokulmasını sağlamak. Ayrıca yeni model arabalar eklendiğinde bridge deseni sayesinde daha basit şekilde sisteme eklenebilecektir.

Factory Deseninin Kullanıldığı Sınıflar

* KiralanacakAraba (Interface)
* Araba
* SporAraba
* SuvAraba
* PickupAraba
* Islemler

**Proxy Deseni**

Proxy deseninde erişim kontrolü sağlayabildiğimiz için kiralama servisini Proxy Deseni ile yapılması daha verimli olacaktır.

***KiralamaProxy*** sınıfı erişim kontolünü sağlayarak daha güvenli ve verimli bir servis olmasını sağlar.

Proxy Deseninin Kullanıldığı Sınıflar

* KiralamaServisi (Interface)
* ProxyKiralama
* KiralamaServisiDemo

**Decorator Deseni**

Kiralanacak arabaya ek hizmet eklenmesini sağlayan en uygun tasarım deseni ***Decorator*** olduğu için ek hizmet ekleme işlemlerini *Decorator* deseni ile yapılacak.

*Decorator* Deseninin Kullanıldığı Sınıflar

* Arac (Interface)
* DecoratorArac (Abstract)
* Araba
* EkMenzilDecorator

**Iterator Deseni**

Araç listesin gizli şekilde tutup erişebilmek için Iterator desenini kullandık.

Iterator Deseninin Kullanıldığı Sınıflar

* Arac (Interface)
* AracListesi
* IteratorArac
* Otomobil

**Strategy Deseni**

Araba kiralama seçenekleri birden fazla olduğu için en uygun tasarım deseni ***Strategy*** tasarım deseni olacaktır.

Strategy Deseninin Kullanıldığı Sınıflar

* KiralamaSecengi(Interface)
* Kirala
* GunlukKirala
* SaatlikKirala

**Visitor Deseni**

Mevcut sınıflara yeni özellik eklememizi gerektirecek durumlarda o sınıflara hiç dokunmadan yeni sınıf oluşturmamızı sağlayan bir tasarım deseni olduğu için Visitor desenini kullanıyoruz.

Visitor Deseninin Kullanıldığı Sınıflar

* Araba (Interface)
* ArabaVisitor (Interface)
* Musteri
* KiralikAraba
* RaporVisitor

**Observer Deseni**

Bildirim gönderme işlemi observer tasarım desenine uyduğu için observer desenini kullandık.

Observer Deseninin Kullanıldığı Sınıflar

* KiralamaServisi (Interface)
* Observer(Interface)
* KiralamaServisiDemo
* OMusteri