**Kullandığımız Teknolojiler**

1. **Flutter**

Flutter, Google tarafından oluşturulan açık kaynaklı bir UI yazılım geliştirme kitidir. Android, iOS, Windows, Mac, Linux ve web için uygulamalar geliştirmek için kullanılıyor.

1. **Node.js**

Node.js, açık kaynaklı, sunucu tarafında çalışan ve ağ bağlantılı uygulamalar için geliştirilmiş bir çalıştırma ortamıdır. Node.js uygulamaları genelde istemci tarafı betik dili olan JavaScript kullanılarak geliştirilir.

1. **MySQL**

MySQL, altı milyondan fazla sistemde yüklü bulunan çoklu iş parçacıklı, çok kullanıcılı, hızlı ve sağlam bir veri tabanı yönetim sistemidir.

1. **Google Cloud Platform ( Places API, Distance Matrix API )**

Google Cloud, Google firmasının Google arama motoru ve Youtube gibi sitelerin de kullandığı sunucu altyapı hizmetlerini son kullanıcıya sunduğu bir bulut bilgi platformudur.

1. **Firebase**

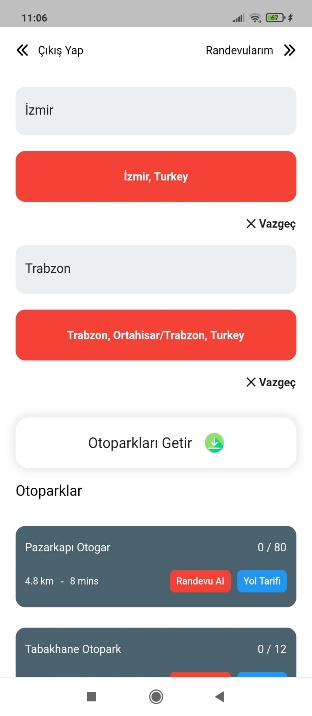
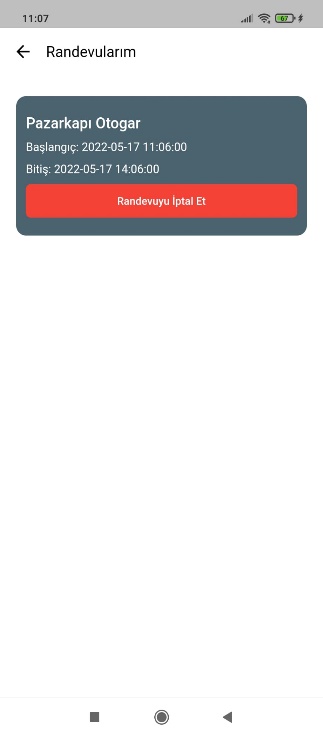
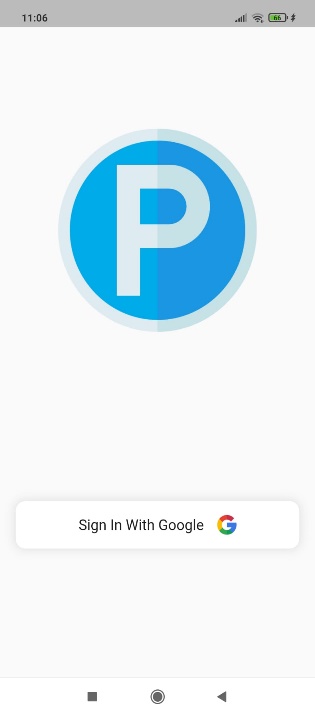
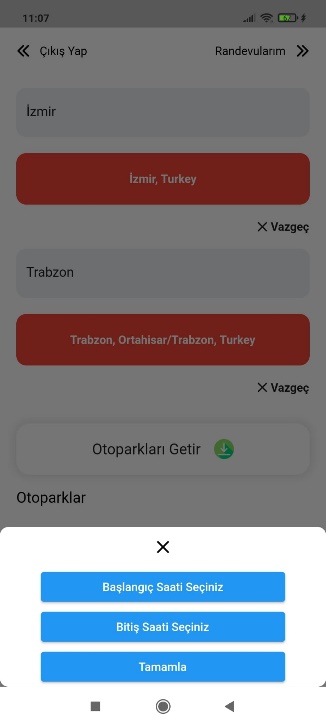
Firebase, Google tarafından mobil ve web uygulamaları oluşturmak için geliştirilmiş bir platformdur.

1. **Linux Server**

Linux Server (Linux Web Sunucusu), açık kaynak kodlu Linux işletim sistemleri üzerine temellendirilen ve içerisinde farklı yazılım bileşeni paketlerini barındırabilen web sunucusudur.

**SÜREÇLER**

1. **Tasarım Süreci**

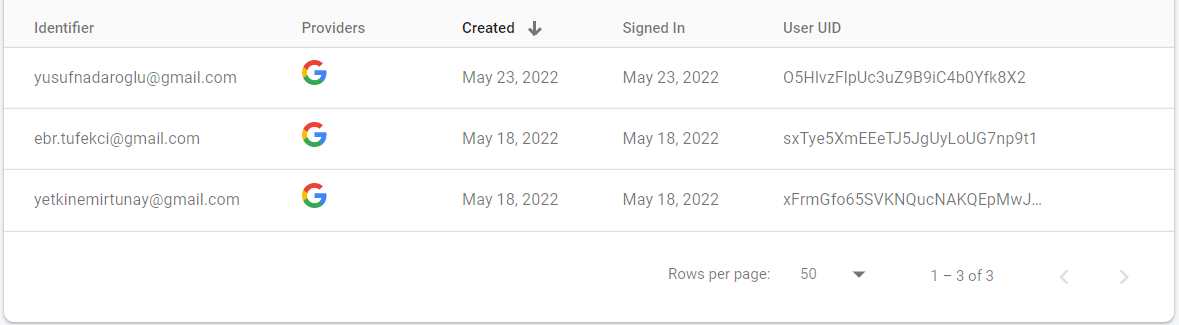
********Tasarım sürecinde ilk başta tasarımını yapacağımız sayfa türlerini belirledik. Ardından diğer uygulamalarda kullanılan tasarımları araştırıp bunlara uygun tasarımları Adobe XD programı üzerinden ayarladık. Tasarımı ayarlarken ana bir renk belirleyip onun üzerinden gittik. Tasarımların sade ve anlaşılır olması en önemli önceliğimizdi. Tasarım sonrası ortaya çıkan görüntüler aşağı kısımda yer alıyor.

1. **Tasarımı Kodlama Süreci**

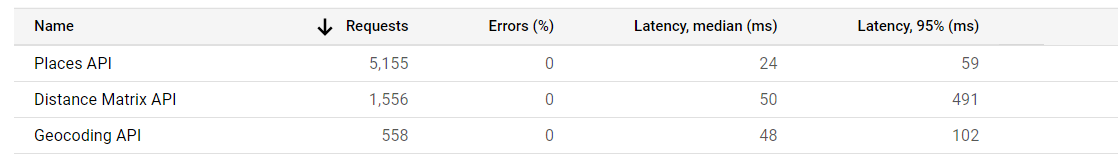
Bu süreç içerisinde hazırladığımız tasarımı en temiz şekilde kodlamaya çalıştık. Bulunan ortak butonlar için tek bir buton kodu yazıp her yerde aynısını kullandık. Logomuzu uygulama içerisinde gösterirken internetten çekmek yerine indirip uygulama içerisine kaydettik. Tasarımdaki iconları genel olarak <https://flaticon.com> sitesinden aldık. Buradaki iconları indirip uygulamamıza ekledik, sürekli internete gidip bunlar için veri harcanmaması için. Randevu alırken aşağıda bir bottom sheet çıkartarak kullanıcının sayfa değiştirmeden kolayca randevu almasını sağladık.

1. **Veri Tabanı Süreci**

Veri tabanı olarak olabildiğince performanslı olması açısından max karakter sınırını ona göre ayarladık. Randevular ve otoparklar adlı iki tablo oluşturduk. Kullanıcı randevu aldığı zaman randevular tablosuna veri ekliyoruz. Daha önceden belirlediğimiz Otoparkları otopark tablosuna kaydettik. Otopark tablosunda id, parking\_name, capacity sütunları bulunurken, Randevu tablosunda id, customer\_id, number\_plate, parking\_id, start\_date, end\_date sütunları bulunuyor. MySQL workbench kullanarak tabloları ve sütunları oluşturduk. MySQL bir çok sunucuda rahat bir şekilde çalışabilen bir veri tabanı sistemi.

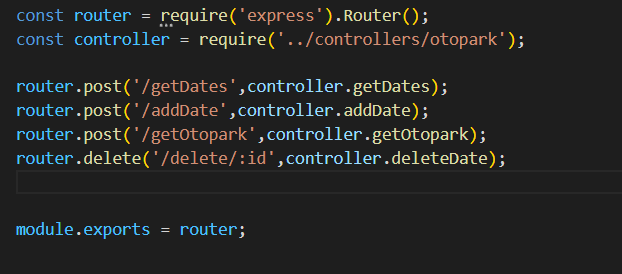
1. **Firebase Süreci**

Firebase ile proje oluşturduktan sonra uygulamayla bağlanması için bazı bilgileri doldurmak gerekiyor. Uygulamanın SHA-1 keyini girdikten sonra projeyle arasında bağlantı kurulmuş oluyor. Flutter ile firebase arasında sürekli bağlantılar yapıldığı için Google bu alanda flutter’da kolayca çalışması için bazı paketleri çıkardı. Firebase kısmında auth ve database kısımları kullanacağımız için firebase\_auth ve cloud\_firestore paketlerini kullandık. Bunun yanı sıra girişin Google ile yapılmasını istediğimiz için de google\_sign\_in paketini kullandık. Firebase’in cloud firestore alanında kullanıcı bilgileri kaydediliyor olup, authentication kısmında da kayıtlı olan kullanıcılar gözüküyor.

1. **Google Cloud Platform Süreci ( Places API && Distance Matrix API)**

Başlangıç ve bitiş noktalarını seçmek için verileri google’dan getirmeye karar verdik ve bunun içinde Google Cloud Platform’un kullanıcılara sunduğu bazı api’lardan yararlandık. Bu apiları kullanmak için bize bir api-key veriliyor ve projemizde bunu yazmamız gerekiyor. Belirli bir kullanıma kadar Google bunların kullanımını ücretsiz olarak bize sunuyor. Otoparklarla bitiş noktası arasındaki mesafeyi bulmak için de bize sunulan Distance Matrix Api’yi kullandık. Arasında ne kadar mesafe olduğunu ve ne kadar sürede gidileceğiyle ilgili bilgileri görmemizi sağlıyor. Başlangıç ve gidilecek yerin enlem ve boylam değerlerini tespit ettikten sonra bunu bizim için yapıyor. Başlangıç noktasını seçtikten sonra enlem ve boylamını bulmak için tekrardan istek atıyoruz.

1. **API Server Süreci**



API oluşturmak için Javascript dilinin Node.js frameworkunu kullandık. Bu kısmın yaptığı şey veri tabanıyla uygulama arasındaki iletişimi sağlamak. Eğer uygulamayla veri tabanı arasında direkt veri alışverişi olsaydı güvenlik açığı olurdu. Bunun için arada bağlantıyı sağlayan bir api server oluşturduk. Api server’ın sürekli açık olması gerekiyor bu yüzden natro üzerinden aldığımız sunucu üzerine bunu kurduk. Bu sayede belirttiğimiz porta istek attığımızda cevap gelebiliyor.

1. **Uygulamayla Server Kısmını Bağlama Süreci**



Uygulamayla server bağlantısını kurmak için http isteği atmamız gerekiyordu. Bunun için flutter kısmında http paketini aktifleştirdik. Uygulamanın çalışma mantığı olarak ilk başta Google girişi ekledik ve kullanıcı giriş yapmadan uygulamaya giremiyor. Kullanıcı girişinden sonra da plakasını istiyoruz. Yine aynı şekilde plakasını girmeden otoparkların gösterildiği sayfaya geçiş yapamıyor kullanıcı. Otoparkları getirmek için sunucuya get isteği atıyoruz. Otoparkların o anki durumunun doluluk oranını göstereceğimiz için otopark isteğine o anki tarihi de iletiyoruz. Başlangıç ve bitiş konumlarını seçmeden otoparkları getir isteği attırmıyoruz çünkü bitiş konumuna yakın otoparkların mesafesinin ve süresinin listelenmesi gerekiyor. Kullanıcı randevu alırken günlük alabiliyor ve isterse bunu iptal edebiliyor. Eğer bir otoparkın doluluk oranı tamamlandıysa otomatik olarak randevu alamıyorsunuz. Kullanıcı istediği zaman çıkış yapıp farklı bir hesapla giriş yapabilir. Her hesabın kendine özel id ve plakası olmak zorundadır.