Mobil Uygulamalar İçin Test Otomasyonu ve RPA Kullanımının İncelenmesi

Yazılım Mühendisliği Ana Bilim Dalı

Dönem Projesi

Erkan Akkoç

Proje Danışmanı: Prof. Dr. Ayşegül Alaybeyoğlu Soy

Ocak 2024

Mobil Uygulamalar İçin Test Otomasyonu ve RPA Kullanımının İncelenmesi

Öz

Bu çalışma, mobil uygulama test otomasyonu ve Robotic Process Automation (RPA) alanlarında kapsamlı bir anlayış sağlamayı amaçlamaktadır. Basit bir uygulamada test otomasyonu için kullanılacak araçların kurulumları ve kullanımlarını açıklayarak, Appium ile test otomasyonuna tabi tutulacaktır. RPA'in iş süreçlerini otomatikleştirme kabiliyetinin mobil uygulama test süreçleriyle nasıl entegre edilebileceğini konusunda proje önerileri sunmaktadır. Ayrıca, yazılım geliştirme süreçlerinde verimliliği artırmak için farklı teknolojilerin nasıl bir araya getirilebileceğini göstererek, endüstrideki mevcut ve gelecekteki projelere değerli bir bakış sunmayı hedeflemektedir.

Anahtar Sözcükler: Mobil uygulama, test otomasyonu, rpa, appium, testng.

Exploration of Test Automation and RPA Utilization for Mobile Applications

Abstract

This study aims to provide a comprehensive understanding in the fields of mobile application test automation and Robotic Process Automation (RPA). It will involve the installation and usage of tools for test automation in a simple application, demonstrating test automation with Appium. The project also proposes ideas on how the automation capabilities of RPA can be integrated with mobile application testing processes. Additionally, it aims to showcase how different technologies can be combined to increase efficiency in software development processes, providing valuable insights into current and future projects in the industry.

Keywords: Mobile application, test automation, rpa, appium, testng

Bu projeyi hazırlama sürecinde desteklerini esirgemeyen ve yanımda olan aileme teşekkür ediyorum.

İçindekiler

Şekiller Listesi

Kısaltmalar Listesi

RPA	Robotic Process Automation
IDE	Integrated Development Environment

Bölüm 1

Giriş

Mobil teknolojinin hızlı evrimi, yazılım geliştirme süreçlerini daha karmaşık hale getirmiş ve bu süreçlere adaptasyonu zorunlu kılmıştır. Günümüzde, yazılım geliştiricileri, kullanıcı dostu ve güvenilir mobil uygulamalar sunabilmek için geliştirme, test ve dağıtım süreçlerinde bir dizi zorluğa karşı karşıyadır. Bu bağlamda, mobil uygulama test otomasyonu ve Robotic Process Automation (RPA), yazılım endüstrisinin bu zorlukları aşma çabalarında önemli roller üstlenmiştir.

Bu projenin temel motivasyonu, mobil uygulama geliştirme süreçlerinde ortaya çıkan karmaşıklığı ele almak ve bu süreçlerde test otomasyonu ile RPA'nın birleşiminden doğan potansiyel avantajları araştırmaktır. Geliştirilen mobil uygulamaların doğru çalışmasını sağlamak, kullanıcı deneyimini artırmak ve yazılım kalitesini güvence altına almak, günümüzdeki rekabetçi pazarda başarılı olmanın anahtarıdır.

Appium gibi güçlü bir mobil test otomasyon aracı kullanarak, mobil uygulama testlerini otomatikleştirmenin yanı sıra, RPA'nın nasıl entegre edilebileceğini ve RPA ile e-posta analizi üzerinden test senaryolarını tetikleme potansiyelini keşfedeceğiz.

Bu proje, mobil uygulama test otomasyonu ve RPA'nın birleşimiyle ilgili birçok soruyu yanıtlamayı hedeflerken, aynı zamanda bu teknolojilerin birleşiminin yazılım geliştirme süreçlerine getirdiği yenilikçi çözümleri ortaya koymayı amaçlamaktadır.

Bölüm 2

Literatür Taraması

Mobil uygulamalar, günlük yaşamımızın ayrılmaz bir parçası haline geldi ve eğlenceden üretkenliğe kadar çeşitli amaçlara hizmet ediyor. Bu uygulamaların kullanılabilirlik ve güvenilirliğini sağlamak, sorunsuz bir kullanıcı deneyimi sunmak açısından hayati önem taşır.

Yazılım testleri, yazılımın eksikliklerini ve potansiyel hatalarını belirleyerek yazılımın kalitesini iyileştirmeyi, güvenilirliğini artırmayı, doğruluğunu ve geçerliliğini sağlamayı amaçlar (Kuday, 2014). Firmalar, ürünlerini belirli standartlara uygun olarak, ürünün amaçlarına yönelik testlere tabi tutarlar. Bu testler, eskiden sadece insan gücüyle manuel olarak gerçekleştirilirken, günümüzde otomasyon testlerine olan ilgi, sağladığı hız ve verimlilik gibi ana nedenlerden dolayı artmaktadır.(Asfaw, 2015)

Mobil uygulamaların test otomasyonu ve RPA'in genel kullanımı konusunda birçok araştırma yapılmıştır. Bu araştırmaların birçoğu, mobil uygulamaların test edilmesi sırasında karşılaşılan zorlukları ele almaktadır. Örneğin, mobil uygulamaların test edilmesi sırasında, diğer uygulamalarla etkileşim, cihazlar üzerindeki ekran, kamera ve diğer donanımlardaki sensörler, donanım ve yazılım platform aileleri, kullanıcı ara yüzleri, enerji tüketimi, iletişim esnasındaki karmaşıklık gibi birçok zorlukla karşılaşılabilmektedir (Takgil ve Kara, 2016).

Mobil uygulamaların test edilmesi sırasında karşılaşılan bu zorlukların üstesinden gelmek için, birçok otomasyon çözümü önerilmiştir. Örneğin, mobil cihazları simüle eden ve tekrarlayan testleri otomatize eden mobil test araçlarının kullanımı, mobil

uygulama testlerindeki en önemli zorluk olan aynı testlerin çok farklı sayıda mobil cihaz üzerinde test edilme ihtiyacını ortadan kaldırmaktadır (2).

Akıllı telefon yazılım uygulamalarının kullanışlılığını test etmek, mevcut olarak karşımıza çıkan bazı zorluklar nedeniyle umut vadeden bir araştırma bağlamıdır. Tek bir mobil özellik, dar bant genişliği, çeşitli çevresel faktörler ve güvenilmez kablosuz veya ağ bağlantısı gibi etkenlerden kaynaklanan bu zorluklar, akıllı telefon yazılım uygulamalarının kullanışlılığını değerlendirme sürecini etkilemektedir. Devam eden özellikler hakkında kullanıcı görüşlerini toplamak için anket ortaya çıkarıldı. Kullanılabilirlik testi anketi sonrasında, akıllı telefon pazarındaki bu iki öncü ürün arasındaki temel farkları belirlemek konusunda yetkin hale geldik (Ahmad, Boota ve Masoom, 2014).

2.1 Appium

Mobil uygulama test otomasyonu, günümüzde yazılım geliştirme süreçlerinde kritik bir rol oynamaktadır. Bu alandaki açık kaynaklı araçlardan biri olan Appium, özellikle mobil uygulama test otomasyonu konusunda popülerlik kazanmıştır. Appium, farklı mobil platformlarda (iOS ve Android) çalışabilen, açık kaynaklı ve cross-platform uyumlu bir otomasyon aracıdır.

2.1.1 Appium'un Temel Özellikleri

2.1.1.1 Cross-Platform Uyumu

Appium, aynı test senaryolarını farklı mobil platformlarda çalıştırabilme yeteneği ile öne çıkar. Bu özellik, geliştiricilere uygulamalarını çoklu platformlarda test etme ve hata tespiti yapma imkanı sağlar.

2.1.1.2 Mobil Cihaz Uyumu

Appium, gerçek cihazlar ve emülatörler üzerinde çalışabilir. Bu, geliştiricilere uygulamalarını farklı ortamlarda test etme ve geliştirme sürecini optimize etme esnekliği sağlar.

2.1.1.3 Açık Kaynak ve Topluluk Desteği

Yazı Appium, açık kaynak bir proje olduğu için geniş bir geliştirici topluluğu tarafından desteklenir. Bu, sürekli güncellenen ve geliştirilen bir araç kullanmanın avantajlarını beraberinde getirir.

2.1.2 Appium Kullanım Senaryoları

2.1.2.1 Fonksiyonel Testler

Appium, mobil uygulamaların temel fonksiyonlarını otomatik olarak test etme yeteneği ile işlevsel testler için ideal bir çözümdür.

2.1.2.2 Performans Testleri

Appium, uygulamaların performansını değerlendirmek ve hata durumlarını tespit etmek için kullanılabilir.

2.1.2.3 CI/CD Entegrasyonu

Appium, sürekli entegrasyon ve sürekli dağıtım (CI/CD) süreçlerine entegre edilebilir, bu da test süreçlerini daha hızlı ve etkili hale getirebilir.

2.1.3 Appium Hakkında Gelecekteki Gelişmeler

Appium ekosistemi sürekli olarak evrim geçirmekte olup, yeni özellikler eklenmekte ve hata gidermeleri yapılmaktadır. Gelecekte, Appium'un daha fazla entegrasyon seçeneği, gelişmiş raporlama araçları ve daha geniş bir cihaz desteği gibi özelliklerin eklenmesi beklenmektedir.

2.2 RPA - Robotic Process Automation

RPA (Robotic Process Automation) kullanımı da mobil uygulamaların test edilmesi sırasında karşılaşılan zorlukların üstesinden gelmek için önerilen bir çözümdür. RPA robotları, web tabanlı, masaüstü veya mobil uygulamalara uyarlanabilir ve yazılımın daha kısa sürede ve maksimum detay ile test edilmesine olanak tanır (3). Bu sayede, iş süreçlerini otomatikleştirip, rutin görevleri insan müdahalesi olmadan gerçekleştirmeyi sağlar. Bu süreçler genellikle tekrarlı ve kurallara dayalıdır, bu nedenle insan gözetimine ihtiyaç duymadan, doğrudan bir yazılım robotu veya "bot" tarafından yürütülebilir.

2.2.1 RPA'in Temel Özellikleri

2.2.1.1 Güçlü Verimlilik ve Hız

RPA, tekrarlanabilir görevleri hızlı ve hatasız bir şekilde gerçekleştirme yeteneği ile bilinir. İnsanlar tarafından yapılan işlemleri daha hızlı ve sürekli bir şekilde gerçekleştirebilir.

2.2.1.2 İnsansız İş Akışı

Belirli iş süreçlerini tamamen otomatikleştirme yeteneği sunar. Bu, insan müdahalesine gerek kalmadan süreçlerin başından sonuna kadar otomatikleştirilebileceği anlamına gelir.

2.2.1.3 Entegrasyon Kabiliyeti

RPA, mevcut sistemlere entegre edilebilir ve genellikle kullanıcıların mevcut altyapılarına dokunmadan işlemleri otomatikleştirmelerine izin verir.

2.2.2 RPA Kullanım Senaryoları

2.2.2.1 Veri Manipülasyonu ve İşleme

Büyük veri setlerinde veri manipülasyonu, filtreleme ve analiz gibi görevleri hızlı bir şekilde gerçekleştirebilir.

2.2.2.2 Fatura İşlemleri

Fatura işleme süreçlerini otomatikleştirerek, fatura verilerini toplama, kontrol etme ve kaydetme görevlerini hızlandırabilir.

2.2.2.3 Müşteri Hizmetleri

Müşteri hizmetleri süreçlerini iyileştirerek, müşteri taleplerini kaydetme, sorgulama ve yanıtlama gibi görevleri otomatikleştirebilir.

2.2.3 RPA'in Avantajları ve Zorlukları

2.2.3.1 Avantajlar

- Hızlı uygulama ve yatırım getirisi (ROI).
- Hata olasılığını azaltma.
- İnsan kaynaklarını daha stratejik görevlere odaklama imkanı.

.

2.2.3.2 Zorluklar

- Karmaşık süreçlerin tam otomasyonu.
- İnsana benzer iş kararları alma yeteneği eksikliği.
- Mevcut sistemlerle uyumsuzluk.

2.3 UiPath

UiPath, Robotic Process Automation (RPA) alanında lider bir yazılım platformudur. Kuruluşlar, UiPath'i kullanarak iş süreçlerini otomatikleştirerek verimliliği artırabilir ve rutin görevleri hızla gerçekleştirebilir. UiPath, kullanıcı dostu bir arayüz, geniş bir kütüphane ve güçlü bir entegrasyon yeteneği ile bilinir.

2.3.1 UiPath'in Temel Özellikleri

2.3.1.1 Kullanıcı Dostu Grafik Arayüzü

UiPath, iş süreçlerini otomatikleştirmek için kullanıcı dostu bir grafik arayüz sunar. Bu, iş analistleri ve geliştiricilerin süreçleri kolayca tasarlamasına ve yönetmesine olanak tanır.

2.3.1.2 Geniş Kütüphane ve Hazır İşlemler

UiPath, kullanıcıların hızlı bir şekilde iş süreçlerini otomatikleştirmelerini sağlayan hazır aktiviteler ve kütüphaneler içerir. Bu, süreçlerin hızlı bir şekilde tasarlanmasını kolaylaştırır.

2.3.1.3 İnsansız İş Akışı

UiPath, süreçleri başından sonuna kadar otomatikleştirme yeteneği ile bilinir. Bu, kullanıcıların süreçlerin tamamen otomatikleştirilmesini sağlayarak insan müdahalesini en aza indirir.

2.3.2 UiPath Kullanım Senaryoları

2.3.2.1 Finansal İşlemler

UiPath, fatura işleme, maliyet hesaplamaları ve finansal raporlamalar gibi finansal süreçleri otomatikleştirmek için kullanılabilir.

2.3.2.2 İnsan Kaynakları

UiPath, işe alım süreçlerini, bordro işlemlerini ve diğer insan kaynakları görevlerini otomatikleştirmek için kullanılabilir.

2.3.2.3 E-posta Analizi ve Uygulama

E-posta içeriğini analiz ederek belirli eylemleri başlatma yeteneği ile iş süreçlerini hızlandırabilir.

2.3.3 UiPath Avantajları ve Gelecekteki Gelişmeler

2.3.3.1 Avantajlar

- Hızlı uygulama ve yatırım getirisi (ROI).
- Kullanıcı dostu arayüz ve hızlı süreç tasarımı.
- Geniş entegrasyon yeteneği.

.

2.3.3.2 Gelecekteki Gelişmeler

UiPath ekosistemi sürekli olarak güncellenir ve yeni özellikler eklenir. Gelecekte, daha gelişmiş yapay zeka (AI) entegrasyonları, genişletilmiş analitik yetenekler ve endüstri spesifik çözümler beklenmektedir.

UiPath, RPA alanında geniş bir etki yaratan bir platform olarak öne çıkarken, kullanıcılarına esneklik, hız ve etkinlik sunarak iş süreçlerini verimli bir şekilde otomatikleştirmelerine olanak tanır.

2.4 Android Studio

Android Studio, Android uygulama geliştirme sürecini kolaylaştırmak ve hızlandırmak için özel olarak tasarlanmış bir entegre geliştirme ortamıdır (IDE).

2.4.1 Android Studio'nun Temel Özellikleri

2.4.1.1 Gelişmiş Kod Düzenleme Araçları

Android Studio, zengin kod düzenleme araçlarına sahiptir ve geliştiricilere hızlı ve verimli bir şekilde kod yazma imkanı sunar. Kod tamamlama, hata denetimi ve kod dökümantasyonu gibi özellikler geliştirme sürecini kolaylaştırır.

2.4.1.2 Gelişmiş Emülatörler ve Aygıt Simülatörleri

Android Studio, geliştiricilere farklı Android cihazlarını simüle etme ve test etme imkanı sağlar. Bu sayede uygulamanın farklı ekran boyutları ve cihaz özelliklerinde nasıl çalışacağını test etmek mümkün olur.

2.4.1.3 Entegre Grafik Kullanıcı Arayüzü (GUI) Tasarım Aracı

Android Studio'nun entegre GUI tasarım aracı, kullanıcı arayüzlerini görsel olarak tasarlamak ve düzenlemek için kullanılabilir. Bu araç, geliştiricilere hızlı prototipler oluşturma ve kullanıcı deneyimini iyileştirme imkanı sağlar.

2.4.1.4 Gelişmiş Derleme ve Dağıtım Araçları

Android Studio, uygulamanın derlenmesi, paketlenmesi ve dağıtılması süreçlerini otomatikleştiren gelişmiş derleme ve dağıtım araçlarına sahiptir. Bu özellikler, uygulamanın hızlı bir şekilde yayınlanmasını sağlar.

2.4.2 Avantajları

Mobil uygulama geliştiricileri için en önemli zorluklardan biri, uygulamalarının farklı cihazlarda ve işletim sistemlerinde nasıl çalıştığını test etmektir. Geleneksel olarak, bu testler için farklı marka, model ve işletim sistemine sahip fiziksel telefonlara ihtiyaç duyulurdu. Ancak Android Studio'nun sunduğu özellikler sayesinde, bu süreç artık çok daha kolay ve verimlidir.

Android Studio, geliştiricilere emülatörler ve sanal cihazlar üzerinde test yapma imkanı sağlar. Bu, geliştiricilerin farklı cihazlarda uygulamalarını test etmelerini sağlar, ancak fiziksel bir cihaza ihtiyaç duymazlar. Özellikle, farklı ekran boyutları, çözünürlükler ve donanım özelliklerine sahip sanal cihazlar oluşturmak, uygulamanın genel kullanılabilirliğini ve performansını test etmek için son derece değerlidir.

Bununla birlikte, Android Studio'nun sağladığı entegre test araçları sayesinde, geliştiriciler uygulamalarının otomatik testlerini de kolayca yapabilirler. Bu, uygulamanın farklı senaryolarda nasıl davrandığını test etmek için önemlidir ve manuel test süreçlerinden kaynaklanan hataları azaltır.

Android Studio'nun mobil test otomasyonu için kullanımı, geliştiricilere zaman kazandırır, test süreçlerini optimize eder ve uygulamanın kalitesini artırır. Farklı marka, model ve işletim sistemlerine sahip fiziksel cihazlara ihtiyaç duymadan,

geliştiriciler uygulamalarını geniş bir yelpazede test edebilirler. Bu da, uygulamanın daha güvenilir ve kullanıcı dostu olmasını sağlar.

2.4.2.1 Appium ve Diğer Test Araçları ile Entegrasyon

Android Studio, Appium gibi popüler test otomasyon araçları ile entegre edilebilir. Bu sayede geliştiriciler, uygulamalarının otomatik testlerini kolayca yapabilir ve test süreçlerini otomatikleştirebilir.

2.4.2.2 Gerçek Cihazlar ve Simülatörlerde Test İmkanı

Android Studio'nun emülatör ve simülatörleri, uygulamaların farklı cihazlarda ve ortamlarda nasıl çalıştığını test etmek için kullanılabilir. Bu, uygulamanın genel performansını ve kullanılabilirliğini değerlendirmek için önemlidir.

2.4.2.3 Hızlı Geri Bildirim ve Hata Ayıklama

Android Studio'nun hata ayıklama araçları, geliştiricilere hızlı geri bildirim sağlar ve uygulamanın performansını ve davranışını canlı bir ortamda izleme imkanı verir. Bu, hata ayıklama sürecini hızlandırır ve uygulamanın kalitesini artırır.

2.4.3 Kullanım Senaryoları

2.4.3.1 Mobil Uygulama Geliştirme

Android Studio, mobil uygulama geliştirme sürecinde en yaygın kullanılan IDE'lerden biridir. Mobil uygulama geliştiricileri, Android Studio'yu kullanarak Java veya Kotlin dillerinde uygulamalarını kolayca geliştirebilirler. Bu senaryoda, Android Studio'nun zengin kod düzenleme araçları ve entegre test araçları sayesinde uygulama geliştirme süreci optimize edilir.

2.4.3.2 Gelişmiş Emülatör ve Simülatör Kullanımı

Android Studio, geliştiricilere farklı cihazlarda ve işletim sistemlerinde uygulamalarını test etme imkanı sağlar. Bu sayede, geliştiriciler uygulamalarının

genel performansını değerlendirebilir ve farklı senaryolarda nasıl çalıştığını görebilirler. Bu senaryoda, farklı cihazlarda uygulama testi yapmak ve uyumluluğu kontrol etmek için Android Studio kullanılabilir.

2.4.3.3 Otomatik Test Senaryoları

Android uygulamaları için otomatik testler oluşturmak ve çalıştırmak için Android Studio'nun entegre araçları mevcuttur. Bu araçlar sayesinde, geliştiriciler uygulamalarını farklı senaryolarda otomatik olarak test edebilir ve hata ayıklama sürecini kolaylaştırabilirler. Bu senaryoda, otomatik testlerin entegrasyonu ve sürekli entegrasyon/teslim (CI/CD) süreçlerinin otomatikleştirilmesi için Android Studio tercih edilebilir.

Bölüm 3

Yöntem

Bu bölümde mobil test otomasyonu için gerekli olan araçların kurulumlarını, nasıl kullanıbileceklerini ve RPA'in bu alanda nasıl kullanılabileceği ile ilgili detayları inceliyor olacağız.

3.1 Kurulumlar

Mobil Test Otomasyonu yapabilmek için bilgisayarımızda yüklü olması gereken bazı uygulamalar vardır.

3.1.1 IDE Nedir ve Eclipse Kurulumu

IDE (Integrated Development Environment), bilişimcilerin yazılım geliştirme sürecini kolaylaştıran bir yazılım aracıdır. IDE'ler, genellikle kod yazma, hata ayıklama, derleme ve dağıtım gibi birçok işlevi tek bir platformda birleştirir. Bu

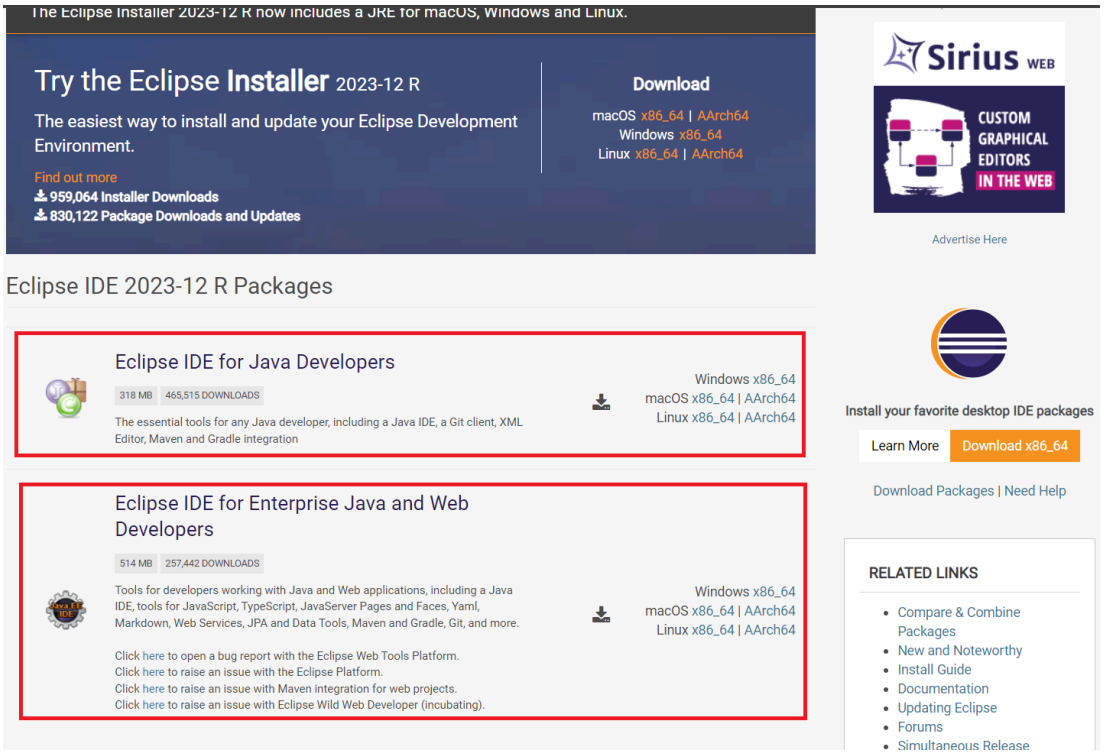
sayede, geliştiricilerin verimliliği artar ve yazılım projeleri daha hızlı bir şekilde geliştirilebilir.

Eclipse, popüler bir Java tabanlı IDE'dir ve genellikle Java geliştirme için tercih edilir. Ancak, genişletilebilir yapısı sayesinde, farklı programlama dilleri ve teknolojiler için de kullanılabilir. Eclipse'in sunduğu özellikler arasında zengin kod düzenleme araçları, hata ayıklama araçları, derleme ve dağıtım araçları bulunur. Ayrıca, Eclipse'in genişletilebilir ekosistemi, kullanıcıların ihtiyaçlarına göre özelleştirilmiş eklentiler eklemelerine olanak tanır.

Android uygulama geliştirme sürecinde, Eclipse sıklıkla tercih edilen bir IDE'dir. Android uygulamaları genellikle Java veya Kotlin dilleriyle geliştirilir ve Eclipse, bu diller için gelişmiş kod düzenleme ve hata ayıklama araçları sunar. Ayrıca, Android SDK ve diğer ilgili eklentileri entegre etmek için kolayca yapılandırılabilir.

Eclipse'in esnek ve kullanıcı dostu arayüzü, geliştiricilere uygulama geliştirme sürecini daha kolay ve verimli hale getirir. Ayrıca, Eclipse'in genişletilebilir yapısı, farklı gereksinimlere ve projelere uyacak şekilde özelleştirilebilir olmasını sağlar.

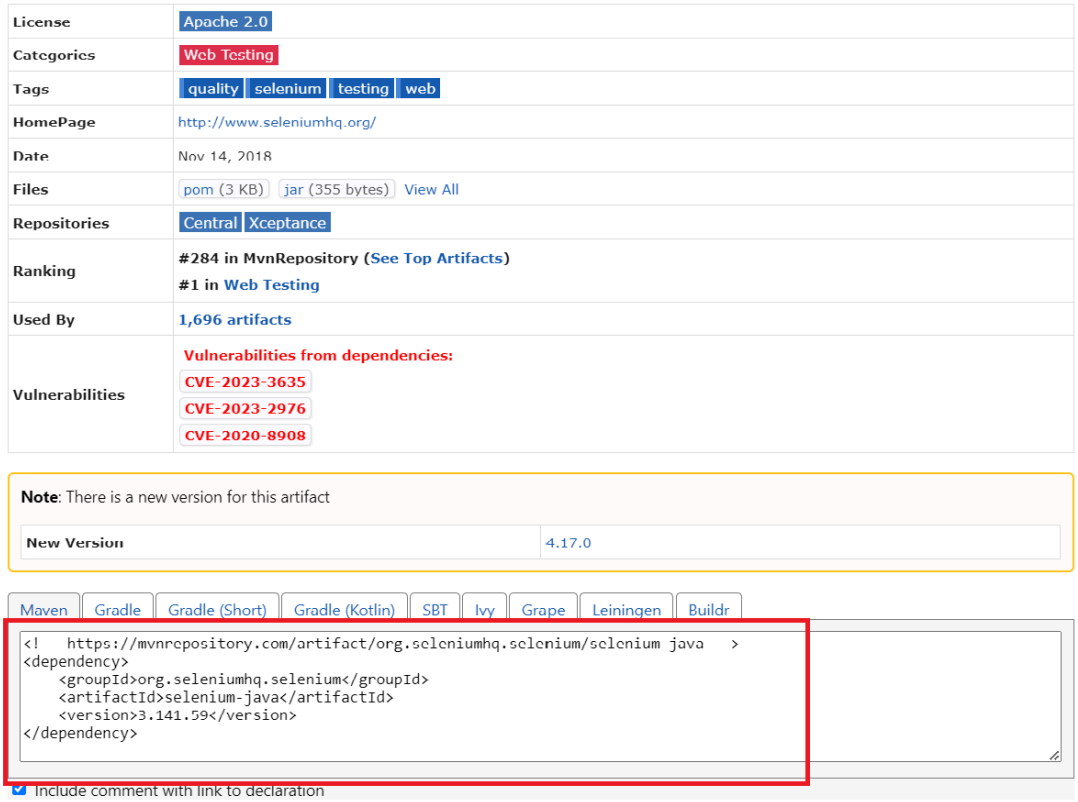
Eclipse’i indirebilmek için kendi sitesinden https://www.eclipse.org/downloads/packages/ adresine giriyoruz ve açılan ekranda "Eclipse IDE for Java Developers" veya “Eclipse IDE for Enterprise Java and Web Developers” sekmesinin sağ tarafındaki alandan kendi işletim sistemimize uygun programı indiriyoruz ve sonrasında basit bir şekilde kuruyoruz.



Şekil 3.1: Eclipse indirme ekranı

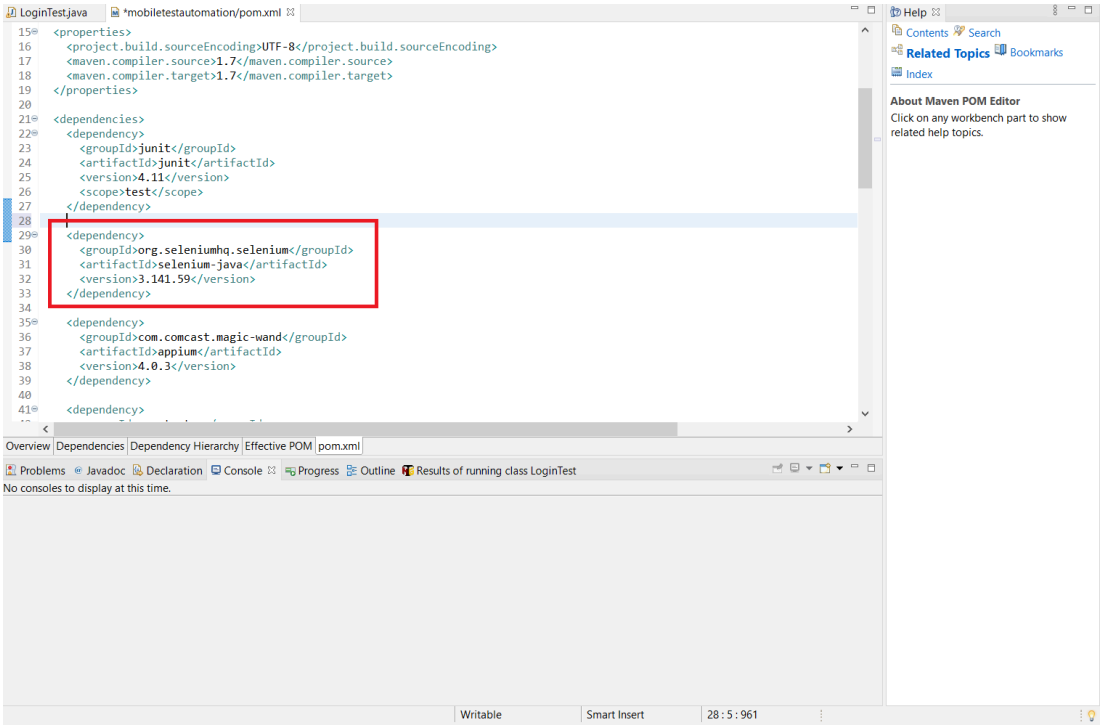
3.1.2 Selenium Kurulumu

Selenium, web uygulamalarının otomatik test edilmesini sağlayan popüler bir açık kaynaklı test otomasyon aracıdır. Selenium'u projenize dahil etmek için öncelikle hangi sürümü kullanmak istediğinize karar vermeniz gerekmektedir. Bunun için Maven projenize kolaylıkla ekleyebilmeniz için https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java sitesine gidip oradan projenizde kullanmak istediğiniz versiyona tıklayınız. Örnek test projemizde 3.141.59 versiyonunu kullanıyor olacağız. Açılan sayfada Maven kısmında yazan kodları kopyalayınız.



Şekil 3.2: Maven repository Selenium kodları

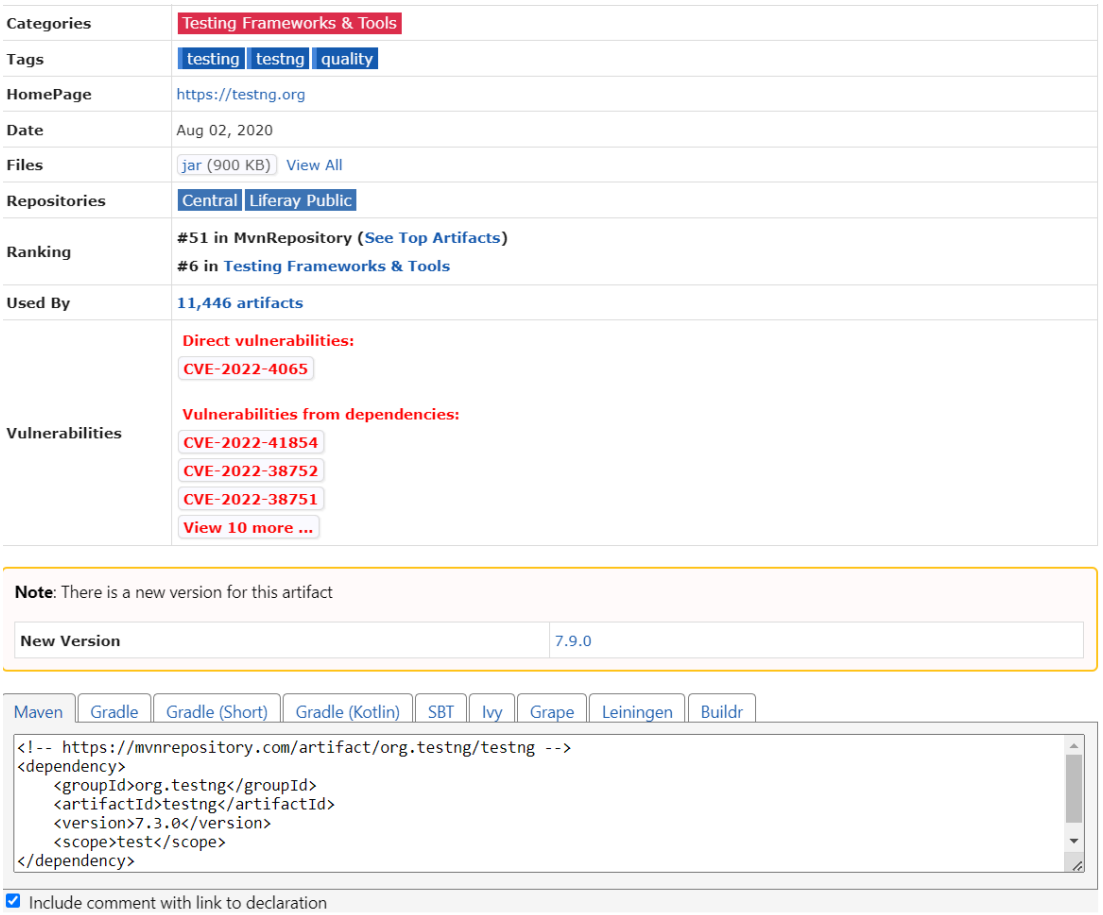
Kopyalamış olduğunuz kodları projeniz içerisindeki pom.xml sayfasının içerisinde bulunan <dependencies> </dependencies> etiketlerinin arasına yapıştırıp Ctrl+S veya Kaydet butonuna tıklamanız gerekmektedir. Bu durumda Eclipse otomatik olarak Selenium'u projenize ekleyecektir.



Şekil 3.3: Selenium - Eclipse pom.xml ekranı

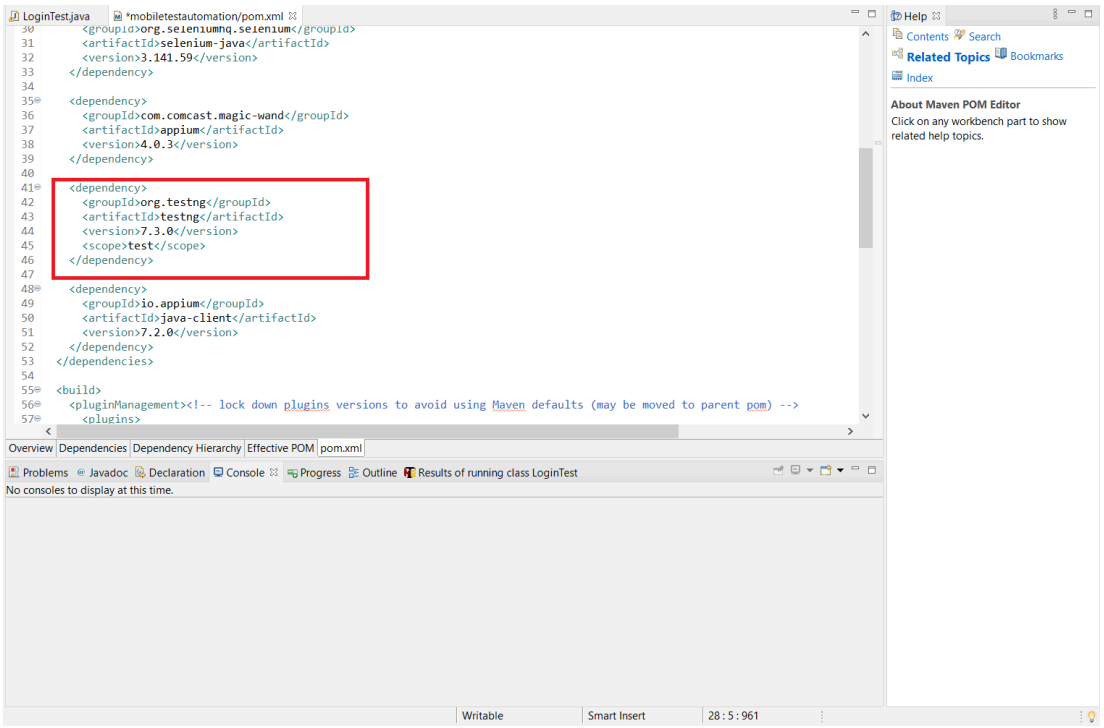
3.1.3 TestNG Kurulumu

TestNG, Java tabanlı testlerin otomatik olarak yürütülmesini ve raporlanmasını sağlayan bir test çerçevesidir. TestNG'yi projenize dahil etmek için https://mvnrepository.com/artifact/org.testng/testng sitesine gidip oradan projenizde kullanmak istediğiniz versiyona tıklayınız. Örnek test projemizde 7.3.0 versiyonunu kullanıyor olacağız. Açılan sayfada Maven kısmında yazan kodları kopyalayınız.



Şekil 3.4: Maven repository Selenium kodları

Kopyalamış olduğunuz kodları projeniz içerisindeki pom.xml sayfasının içerisinde bulunan <dependencies> </dependencies> etiketlerinin arasına yapıştırıp Ctrl+S veya Kaydet butonuna tıklamanız gerekmektedir. Bu durumda Eclipse otomatik olarak TestNG'yi projenize ekleyecektir.

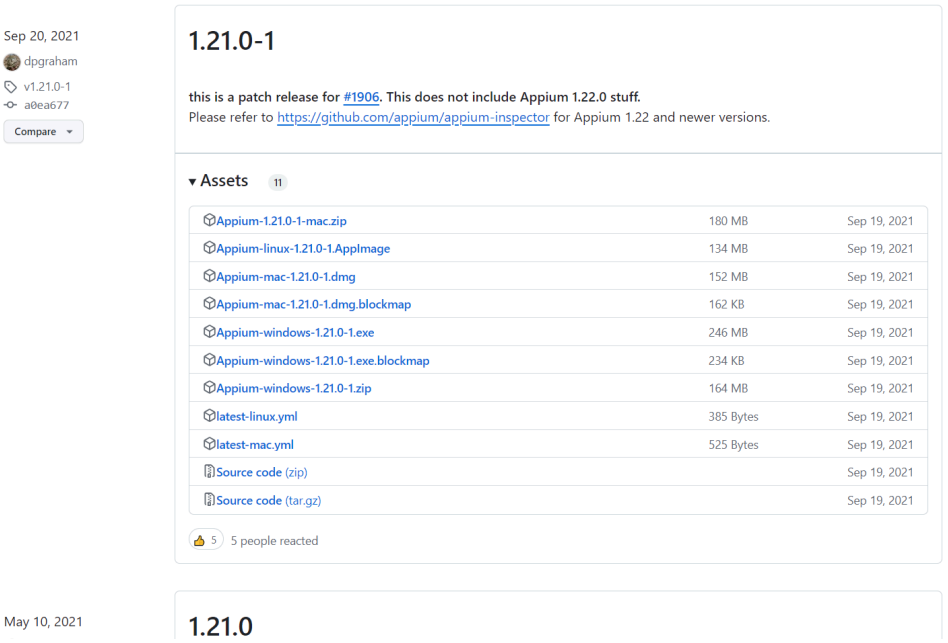


Şekil 3.5: TestNG - Eclipse pom.xml ekranı

3.1.4 Appium Kurulumu

Appium, mobil uygulamaların otomatik test edilmesini sağlayan bir açık kaynaklı test otomasyon aracıdır. Appium'u projenizde kullanabilmek için ikisi pom.xml'e eklenmek üzere toplam üç farklı kurulum işlemi yapmanız gerekmektedir. Hem masaüstü olan versiyonunu bilgisayarınıza kurmanız hem de proje kodlarınıza eklemeniz gerekmektedir.

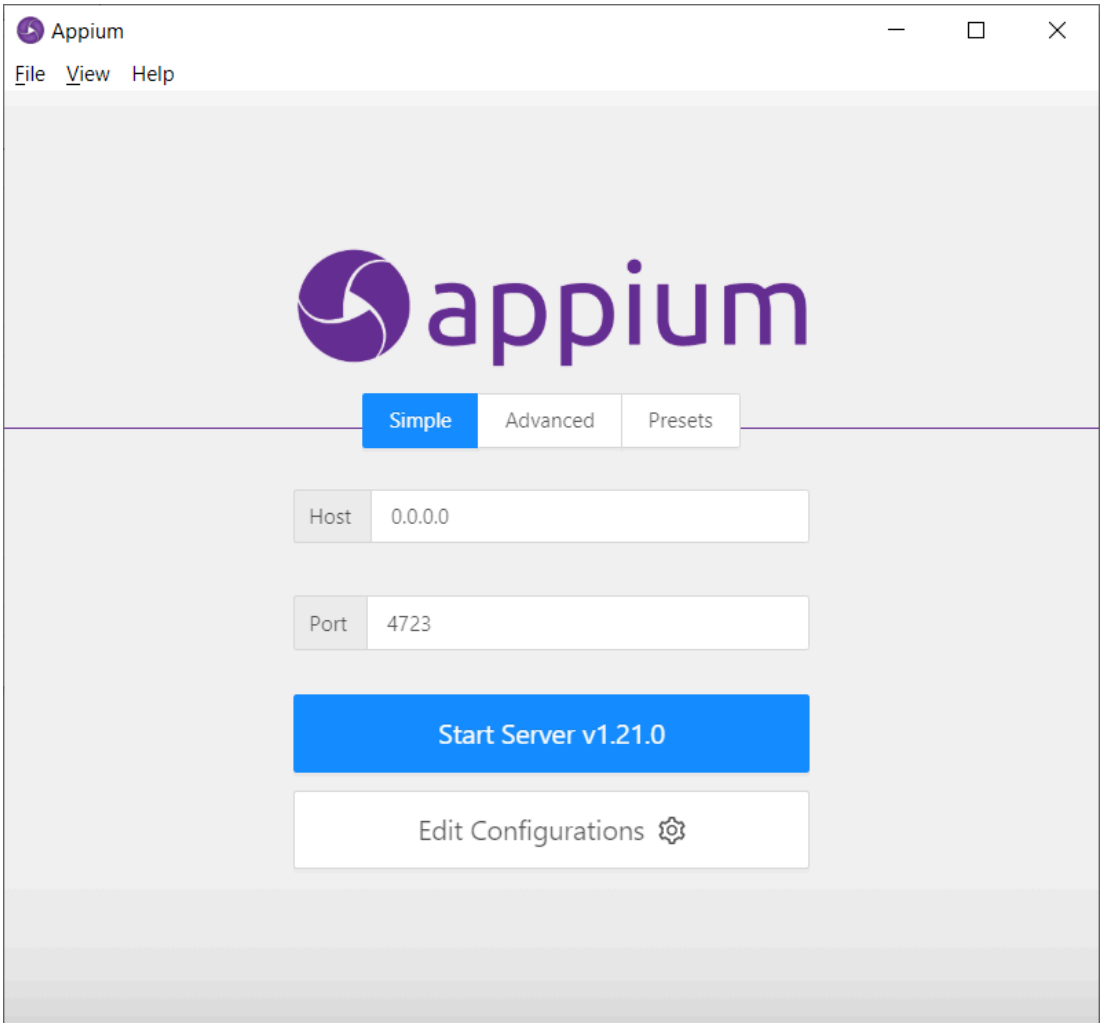
Öncelikle Appium'un masaüstü uygulamasını indirmek için https://github.com/appium/appium-desktop/releases adresine gitmeniz gerekmektedir. Bu sayfada Appium'un tüm sürümleri listelenmektedir. Sayfanın üstünde gösterilen, işletim sisteminize uygun olan en son sürümü indirmeniz tavsiye edilir.. Tabi siz istediğiniz uygun sürümü de kurabilirsiniz. Örnek projemizde 1.21.0 versiyonu kullanılacaktır.



Şekil 3.6: Appium indirme dosyası

İndirdiğiniz dosyayı çift tıklayarak çalıştırdıktan sonra karşınıza basit bir kurulum ekranı gelecektir. Sizin için uygun olan seçenekleri seçip hızlıca kurulumu tamamlayabilirsiniz.

Kurulum işlemi bittikten sonra uygulamayı çalıştırmanız gerekmektedir.



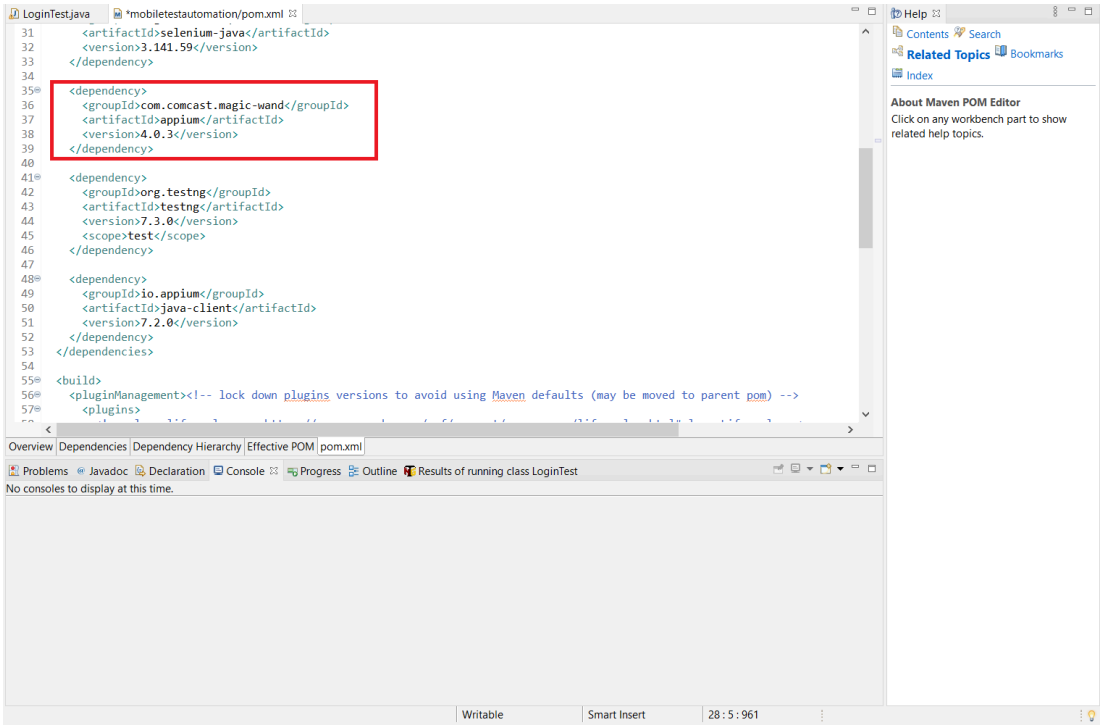
Şekil 3.7: Appium açılış ekranı

Bu ekranda Start Server v1.21.0'a tıklayarak Appium'un çalışmasını başlatabilirsiniz.

Yapmanız gereken diğer iki işlem ise Appium ve Java Client'ın pom.xml'e eklenmesi olacaktır. Bunun için öncelikle Appium'u ekliyoruz. https://mvnrepository.com/artifact/com.comcast.magic-wand/appium sitesinde 4.0.3 versiyonuna tıklıyoruz. Açılan sayfada Maven kısmında yazan kodları kopyalayınız.

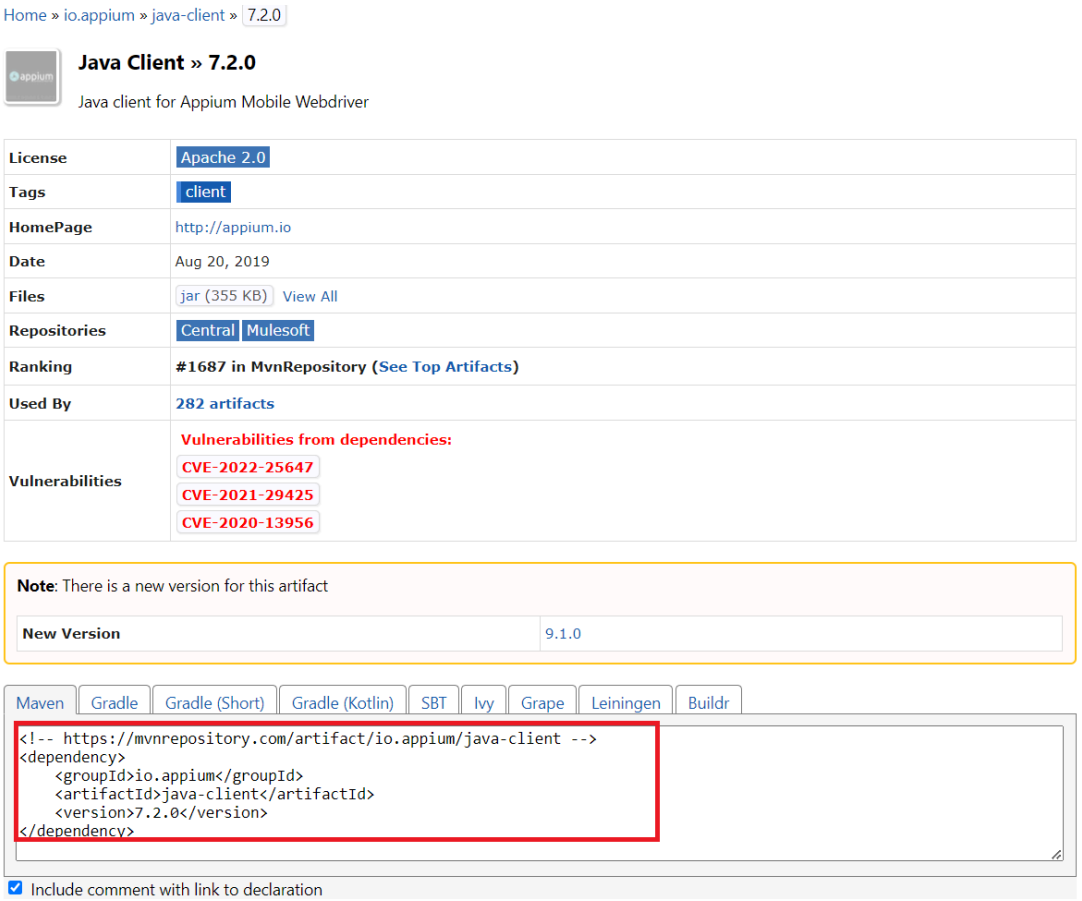
Şekil 3.8: Maven repository Appium kodları

Kopyalamış olduğunuz kodları projeniz içerisindeki pom.xml dosyasının içerisinde bulunan <dependencies> </dependencies> etiketlerinin arasına yapıştırınız. Henüz kaydetmeyiniz çünkü Java Client'ı da aynı şekilde buraya ekleyeceğiz. O işlemi tamamladıktan sonra kaydetme işlemini yapacağız.



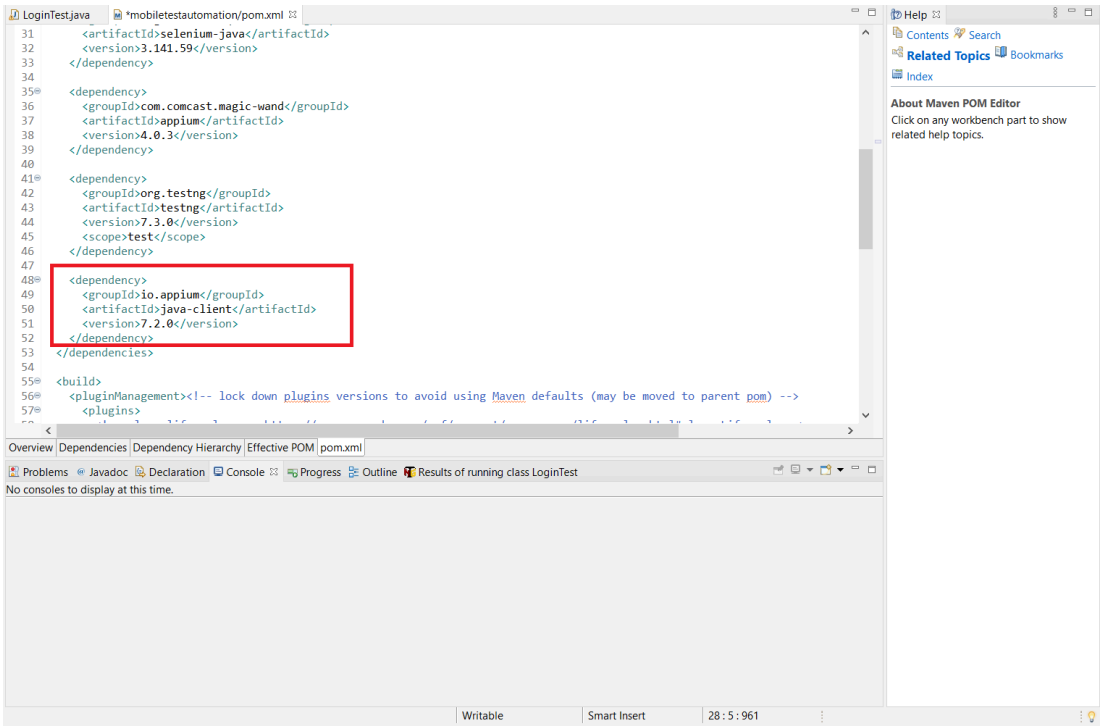
Şekil 3.9: Appium- Eclipse pom.xml ekranı

Appium'u ekledik ve artık Java Client'ın pom.xml'e eklenmesi gerekiyor. Bunun için https://mvnrepository.com/artifact/io.appium/java-client sitesine gidip projemize uygun olan versiyona tıklıyoruz. Örnek projemizde 7.2.0 versiyonuna kullanıyor olacağız. Açılan sayfada Maven kısmında yazan kodları kopyalayınız.



Şekil 3.10: Maven repository Java Client kodları

Kopyalamış olduğunuz kodları projeniz içerisindeki pom.xml sayfasının içerisinde bulunan <dependencies> </dependencies> etiketlerinin arasına yapıştırıp Ctrl+S veya Kaydet butonuna tıklamanız gerekmektedir. Bu durumda Eclipse otomatik olarak Appium ve Java Client'ı projenize ekleyecektir.

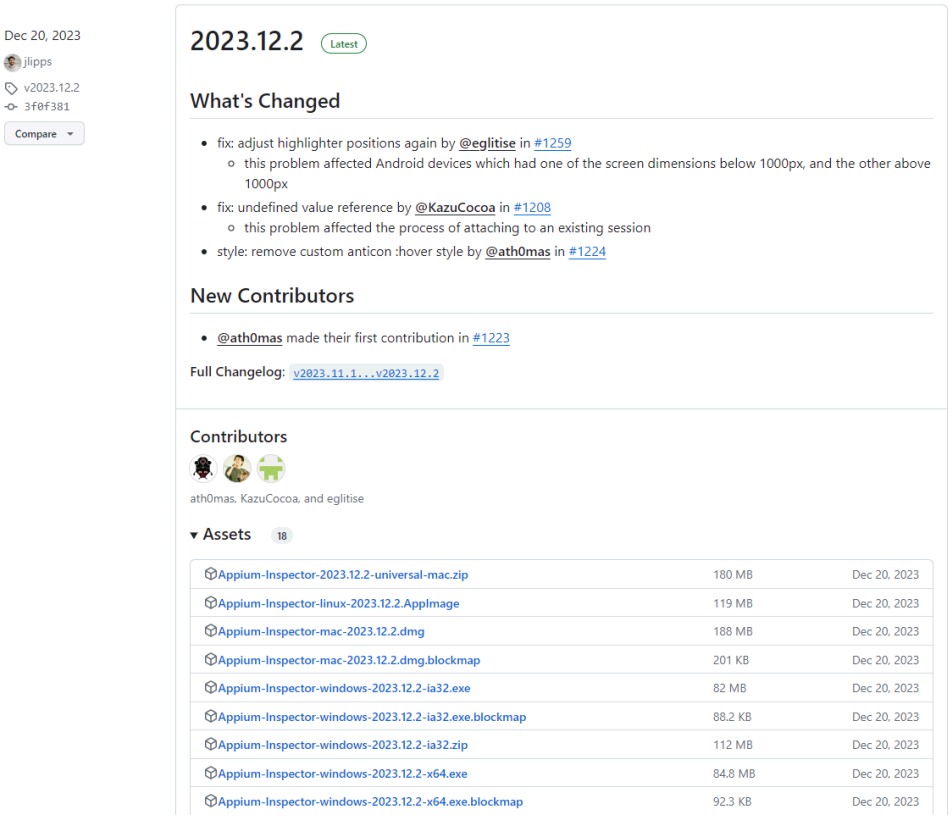


Şekil 3.11: Java Client - Eclipse pom.xml ekranı

Artık Appium ile ilgili dosyalar bilgisayarınıza ve projenize başarıyla kurulmuş bulunmaktadır.

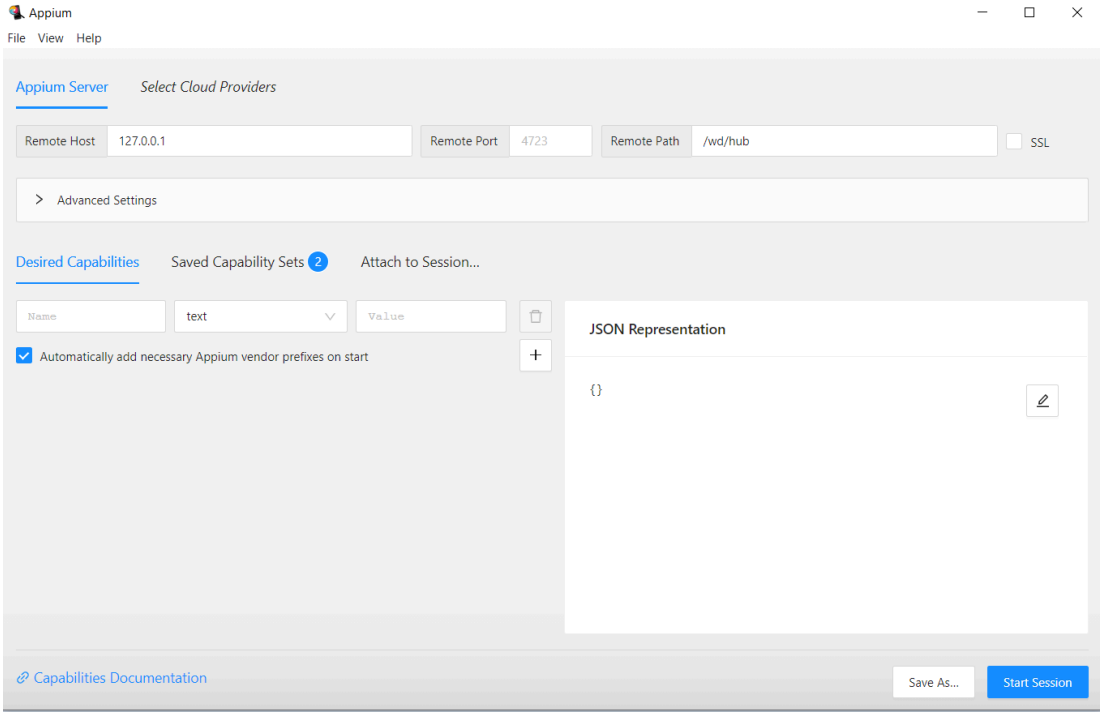
3.1.5 Appium Inspector Kurulumu

Appium Inspector, mobil uygulamaların otomatik test edilmesini sağlayan bir araçtır ve geliştiricilere uygulamanın UI bileşenlerini tanımlamak ve test senaryoları oluşturmak için yardımcı olur. Appium Inspector'u indirmek için https://github.com/appium/appium-inspector/releases sitesine gidip, kullanmak istediğiniz versiyonun işletim sisteminize uygun olan halini indirmeniz gerekmektedir.



Şekil 3.12: Appium Inspector indirme ekranı

İndirdiğiniz dosyayı çalıştırdığınız anda otomatik olarak uygulama açılacaktır. Ekstra bir kurulum işlemi yapmanız gerekmeyecektir.



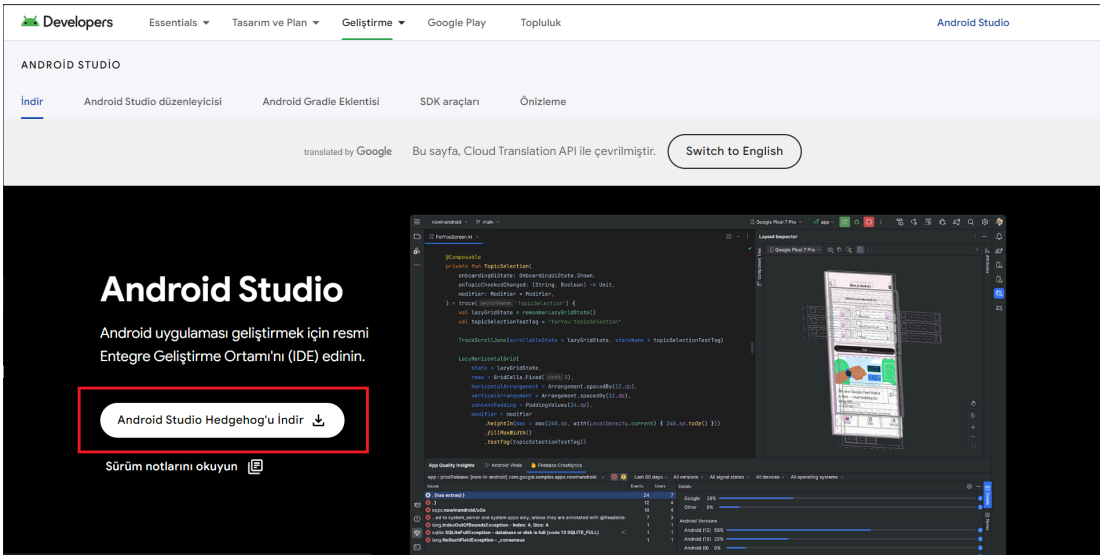
Şekil 3.13: Appium Inspector ana sayfası

Start Session butonuna tıkladığınız zaman hata almamanız için Remote Host bilginizin Appium uygulamasınıyla aynı host adresi olduğuna ve Appium Server'ı başlattığınızdan emin olmanız gerekmektedir, aksi taktirde Appium Inspector uygulaması hata verecektir.

3.1.6 Android Studio Kurulumu

Android Studio, Android uygulama geliştirme için resmi entegre geliştirme ortamıdır. Android Studio'nun en son sürümünü indirerek ve kurarak Android uygulamaları geliştirmeye başlayabilirsiniz.

Android Studio'yu resmi web sitesinden ücretsiz bir şekilde indirebilirsiniz. Tarayıcınızı açın ve https://developer.android.com/studio adresine gidin. "Android Studio Hedgehog'u İndir" düğmesine tıklayarak indirme işlemine başlayabilirsiniz.

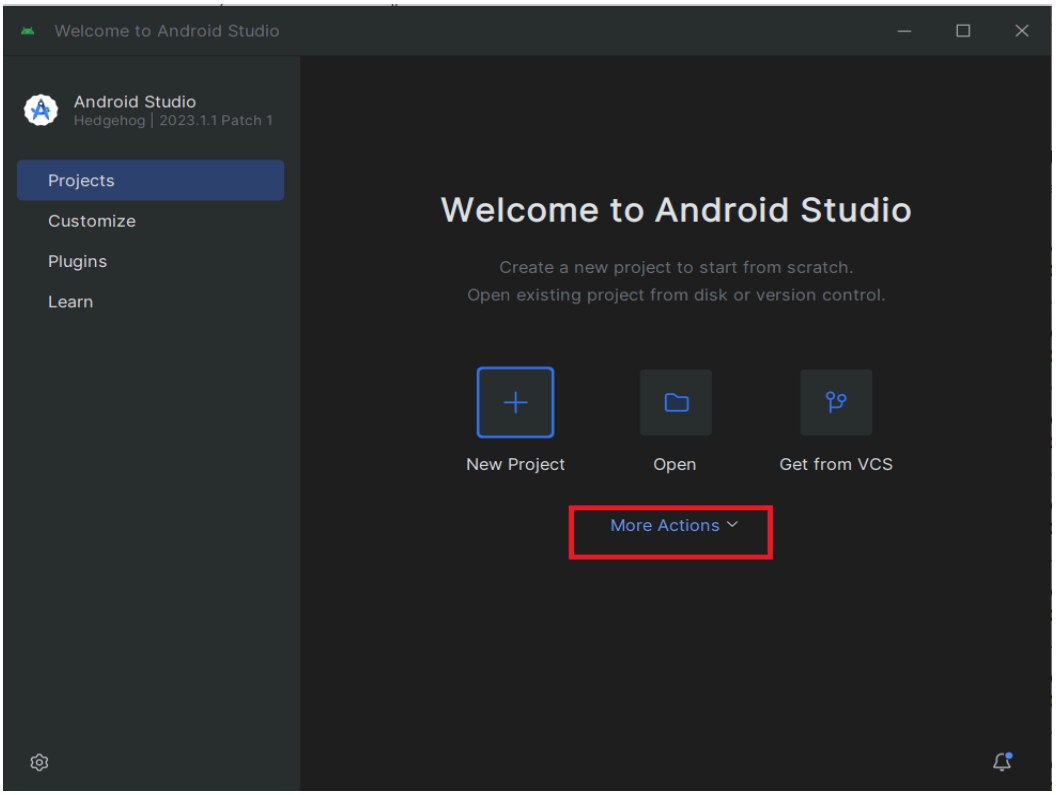


Şekil 3.14: Android Studio web sitesi

İsterseniz https://developer.android.com/studio/archive?hl=tr adresine giderek işletim sisteminize uygun olan farklı Android Studio versiyonlarını indirebilirsiniz. İndirdiğiniz dosyayı çalıştırdığınız takdirde Android Studio uygulaması hemen açılacaktır.

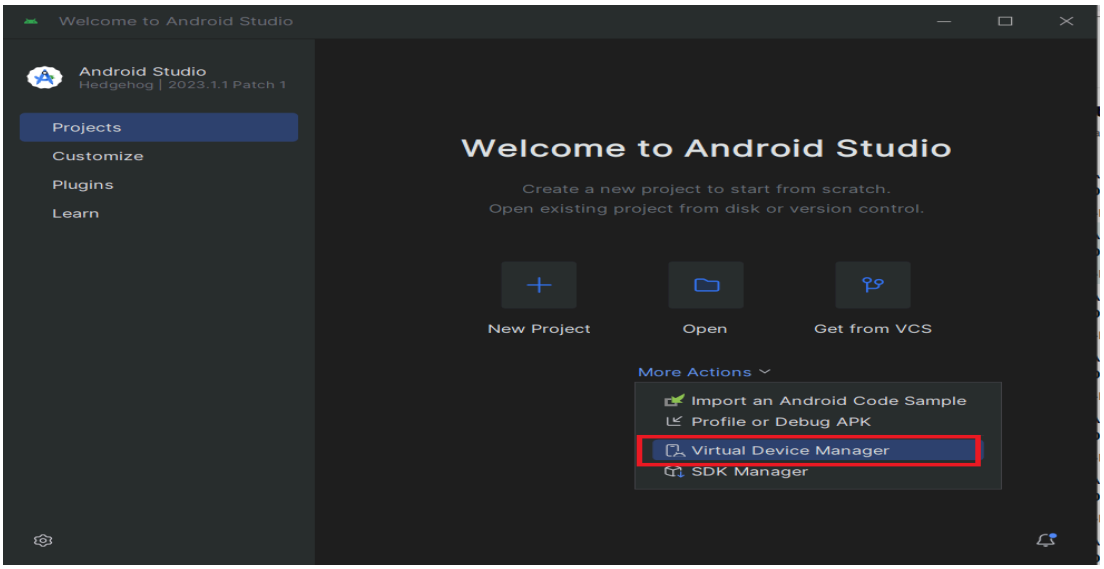
3.1.7 Emülatör'ün Hazırlanması

Android Studio içerisinde yer alan emülatörü hazırlamak için Android Studio uygulamasını çalıştırmanız gerekmektedir. Çalıştırdıktan sonra "More Actions" butonuna tıklayınız



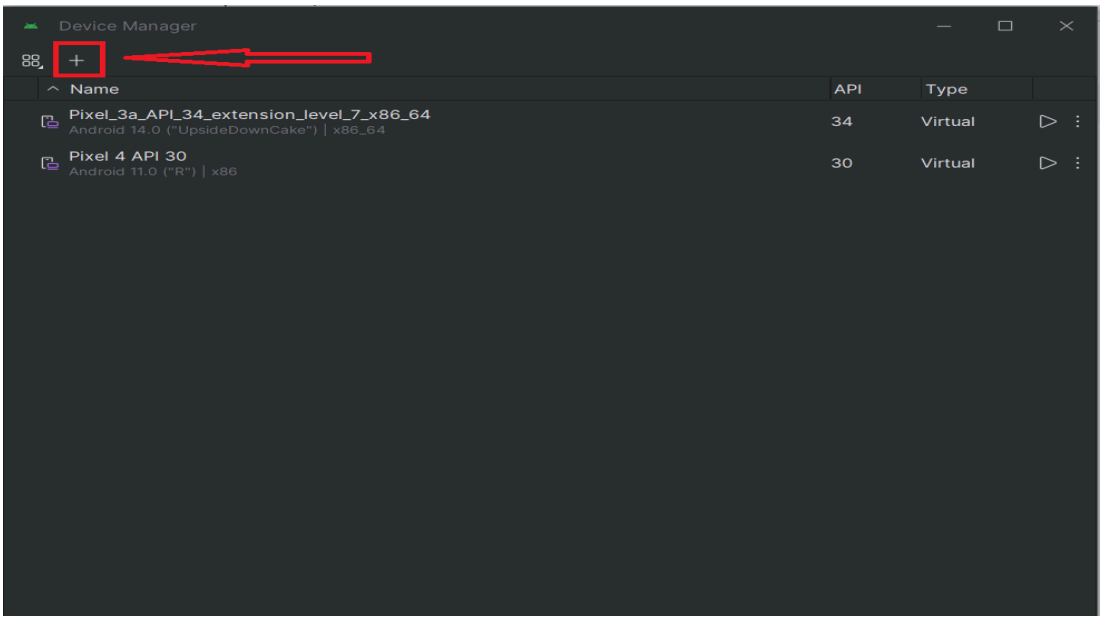
Şekil 3.15: Android Studio ana ekran

More Actions kısmının içerisinde bulunan "Virtual Device Manager" butonuna tıklayınız.



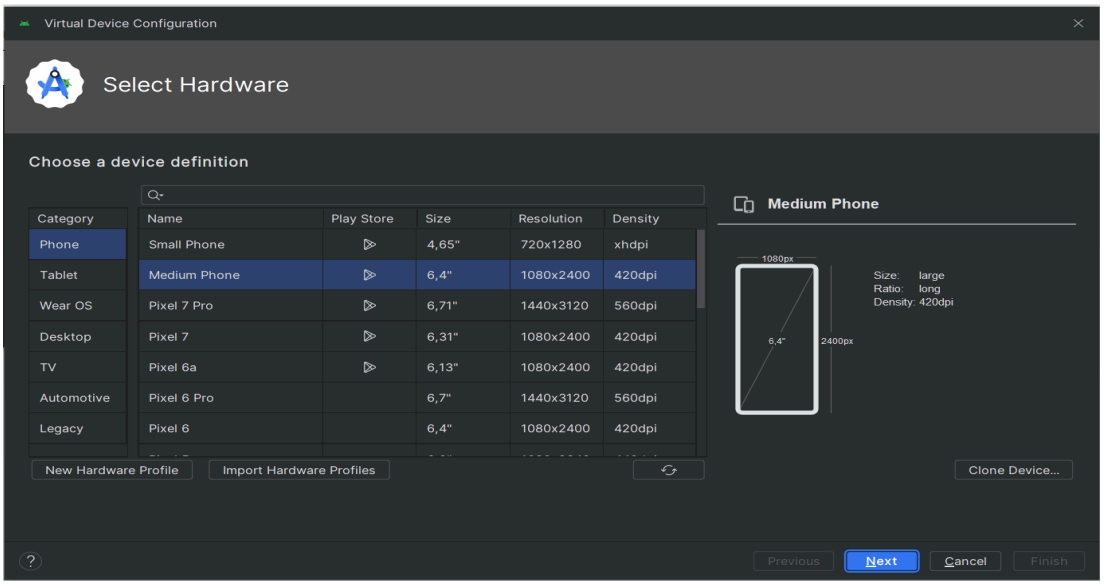
Şekil 3.16: Android Studio - more actions

Device Manager sayfası karşınıza geldiği zaman “+” butonuna tıklayarak yeni bir cihaz oluşturabilirsiniz.



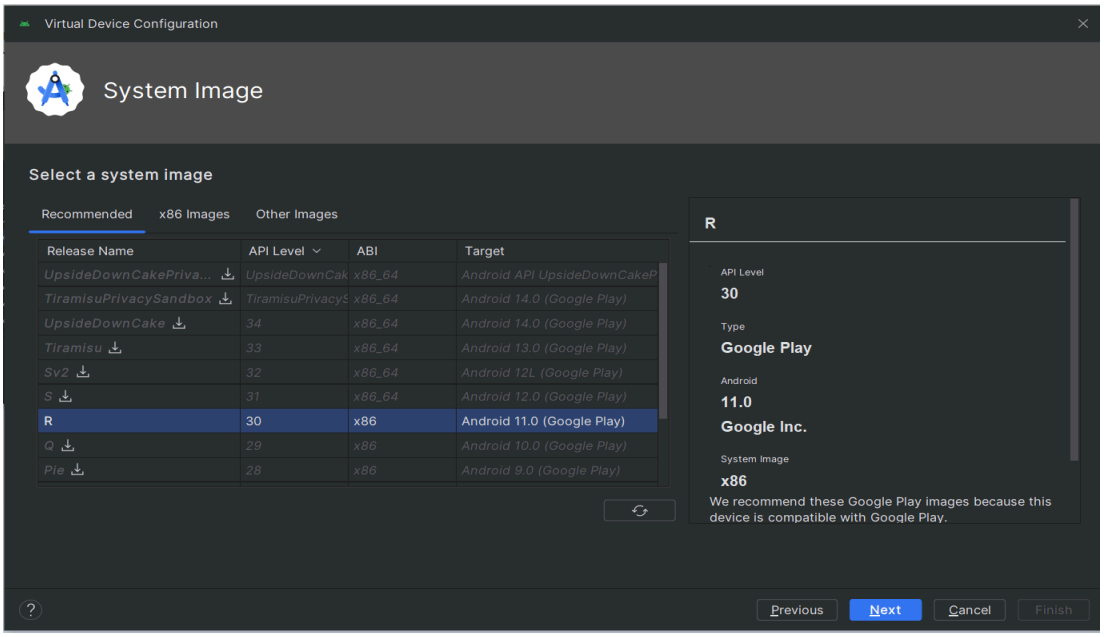
Şekil 3.17: Device Manager

Karşımıza gelen sayfadan test etmek istediğimiz cihazı seçip “Next” butonuna tıklıyoruz.



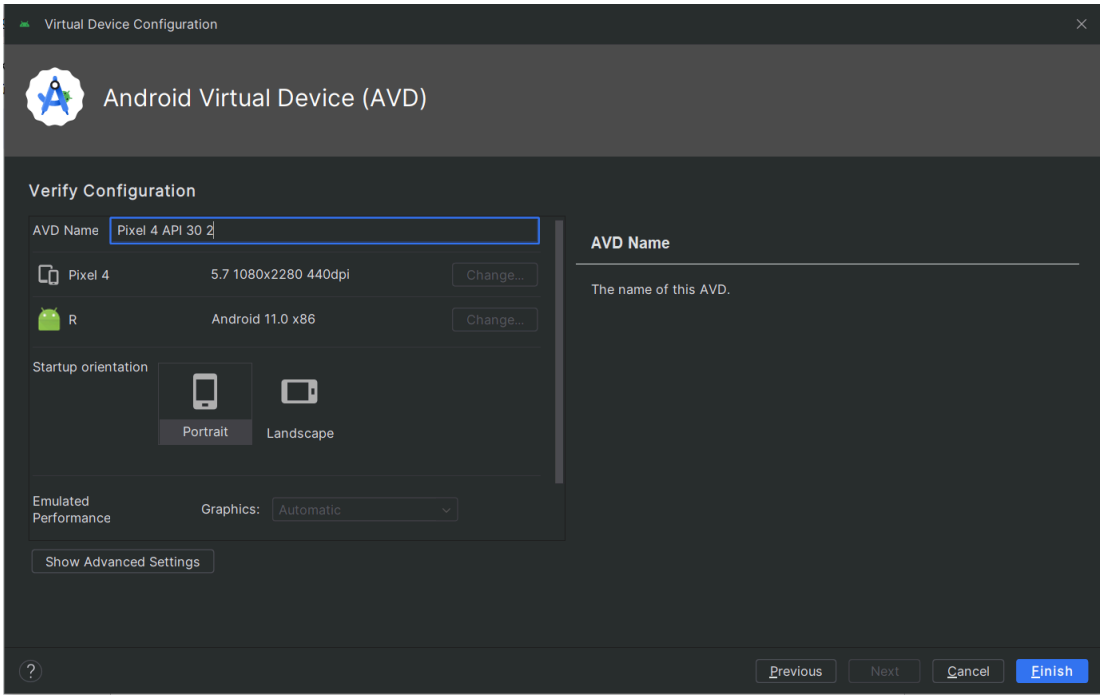
Şekil 3.18: Virtual Device Configuration

Karşımıza gelen ekranda sol tarafta Android sürümleri yer alıyor. Yanında aşağı doğru ok işareti olanlar henüz indirmemiş olduğunuz Android sürümleridir. O butonlara tıklayarak istediğiniz Android sürümünü indirebilir ve bu sayede oluşturacağınız emülatörünüzde kullanabilirsiniz. İstediğiniz versiyonu seçtikten sonra "Next" butonuna tıklayınız.



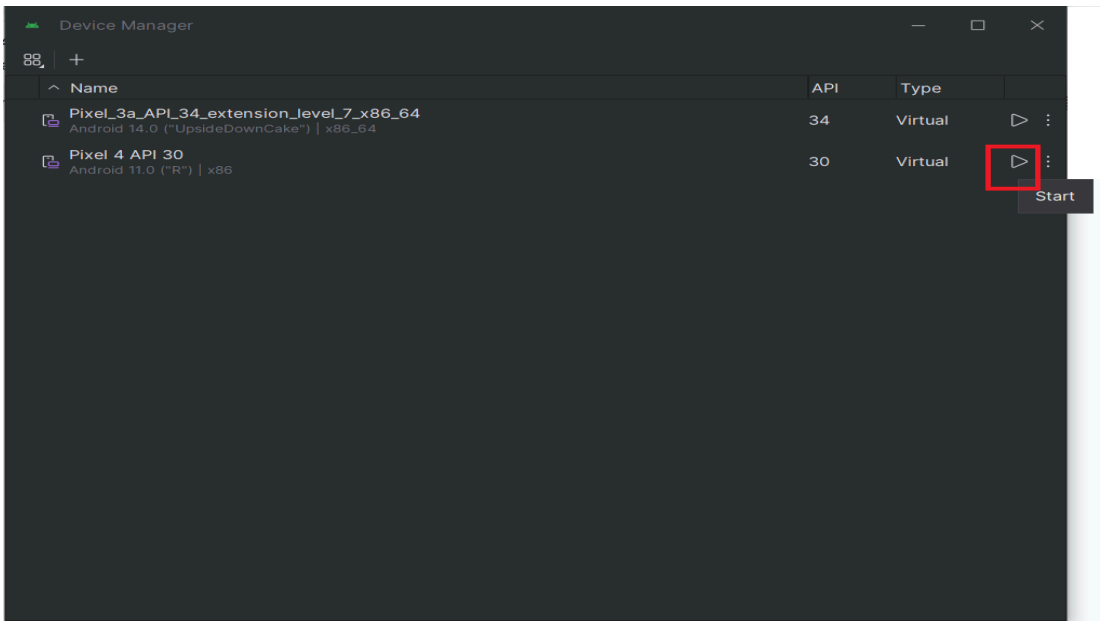
Şekil 3.19: Virtual Device Configuration - Android versiyonu

Emülatörün kurulmasının son aşamasında cihazımıza isim verebilir, açılışta yatay veya dikey açılacağı belirlenebilir.



Şekil 3.20: Virtual Device Configuration - Emülatör adı

Seçimlerinizi yaptıktan sonra “Finish” butonuna tıkladığınız zaman Emülatörünüz kurulacaktır. Emülatörü çalıştırmak için emülatör isminin sağında bulunan “Play” tuşuna tıklayınız. Butonun üzerine geldiğiniz zaman orada “Start” yazacaktır.

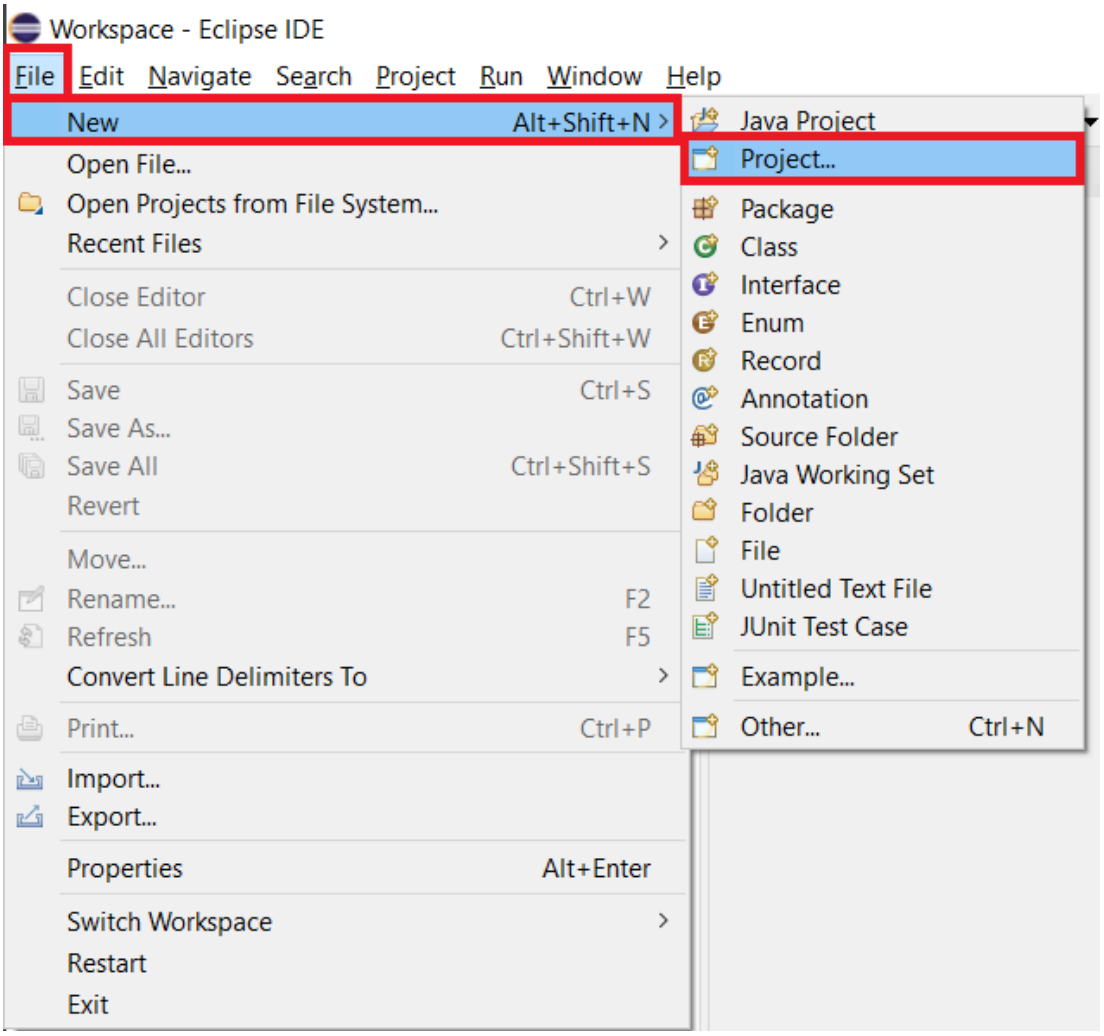


Şekil 3.21: Device Manager - Emülatör çalıştırma

3.2 Örnek Proje

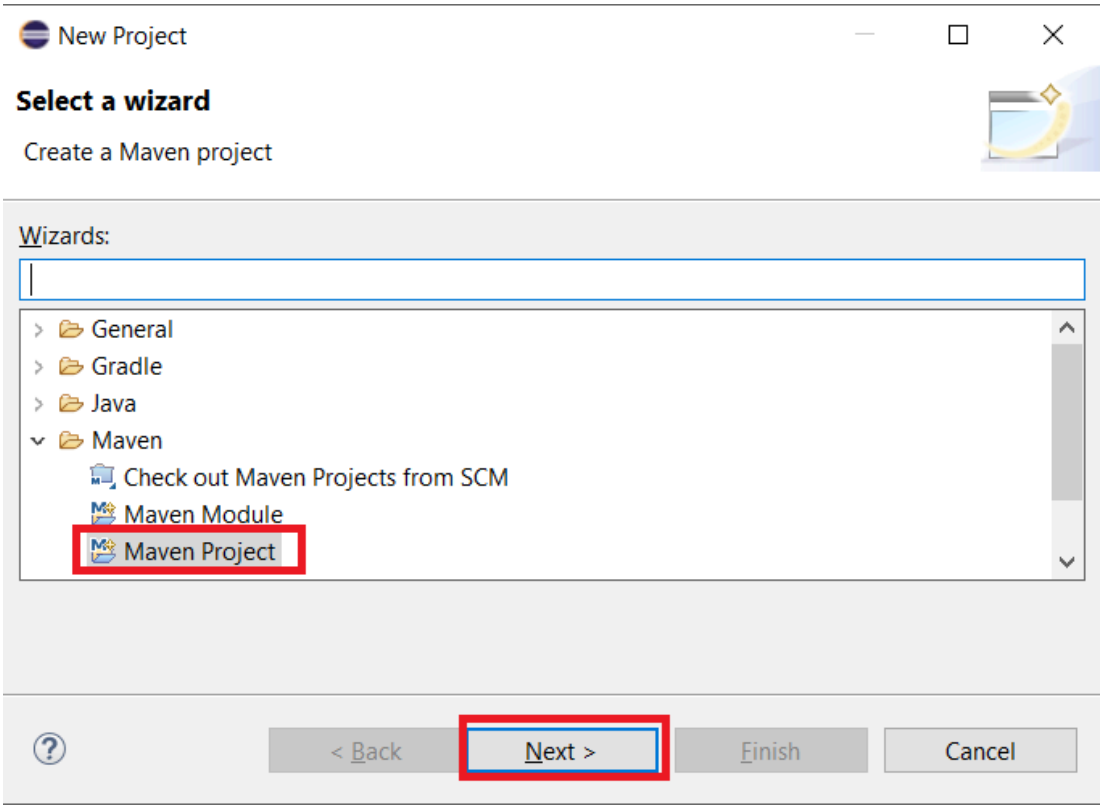
3.2.1 Maven Projesinin Oluşturulması

Örnek projenin yapılabilmesi için öncelikle Eclipse üzerinden bir Maven projesi oluşturulması gerekmektedir. Bunun için File -> New -> Project veya "Alt+Shift+N" tuşlarına aynı anda basarak karşımıza gelen alandan Project'i seçmemiz gerekmektedir.



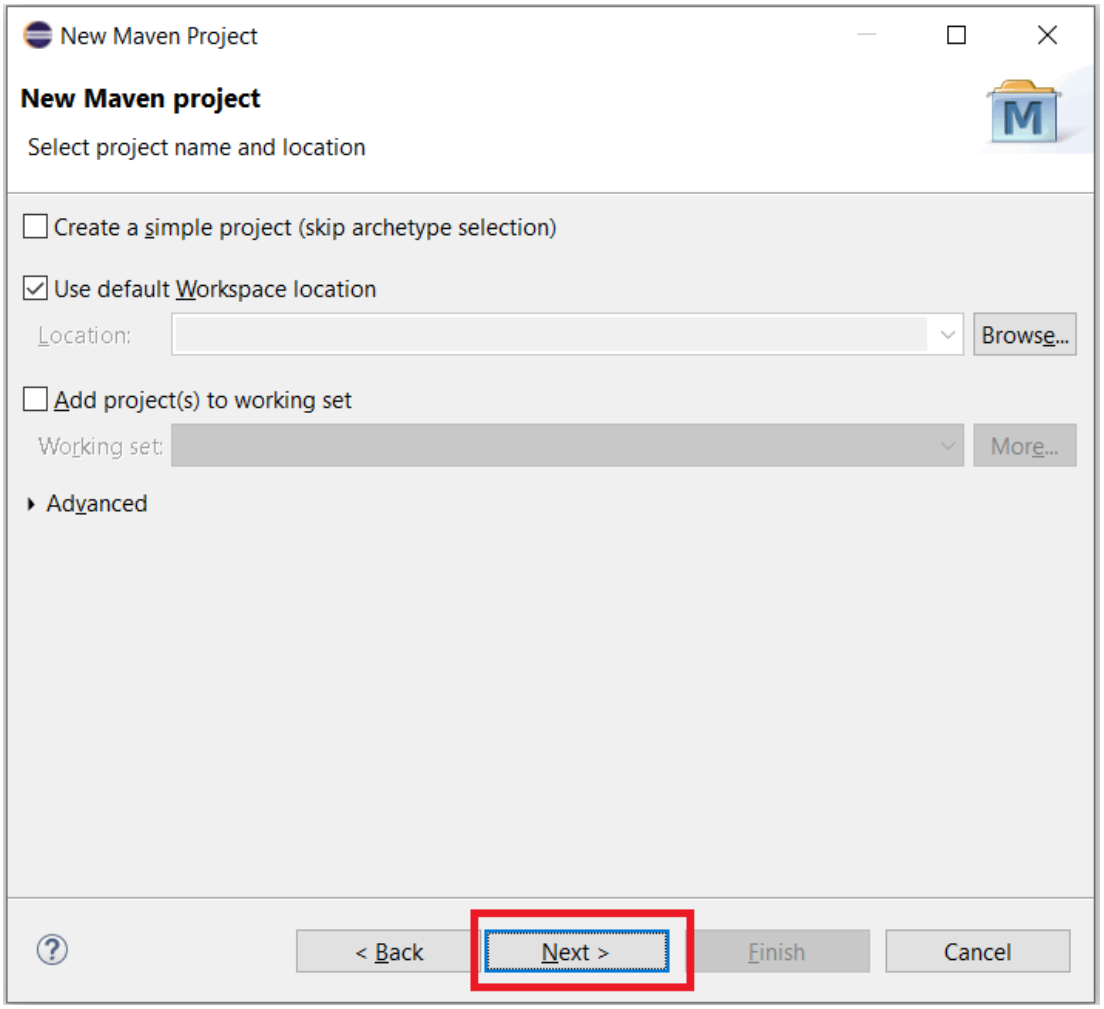
Şekil 3.22: Yeni Proje Oluşturma

Karşımıza projemizin türünü seçebileceğimiz ekran gelecek. Oradan "Maven Project"i seçip, "Next" butonuna tıklıyoruz.



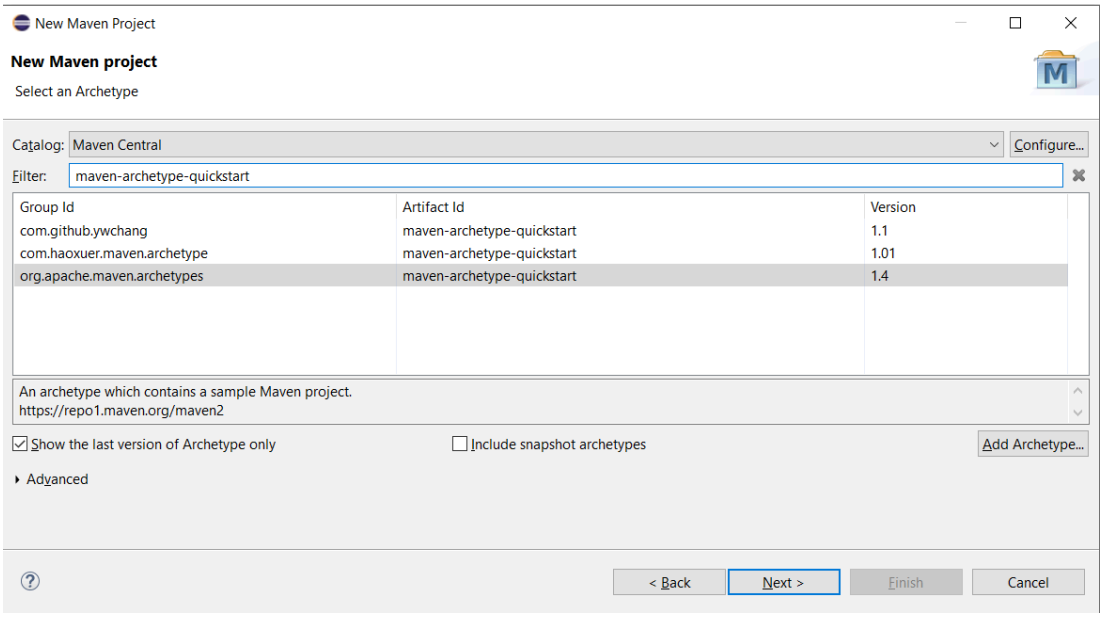
Şekil 3.23: Maven Project Seçimi

Eğer Maven projesi ile ilgili bilgileri girmek istenmiyorsa, “Create a simple project” seçeneği seçilebilir. Fakat bu bilgiler girilmek istenirse resimde olduğu gibi o alan boş bırakılabilir. “Next” butonuna tıklayarak bir sonraki ekrana geçilir.



Şekil 3.24: Çalışma alanının belirlenmesi

Archetype olarak “maven-archetype-quickstart”ı seçip “Next”e tıklanır.

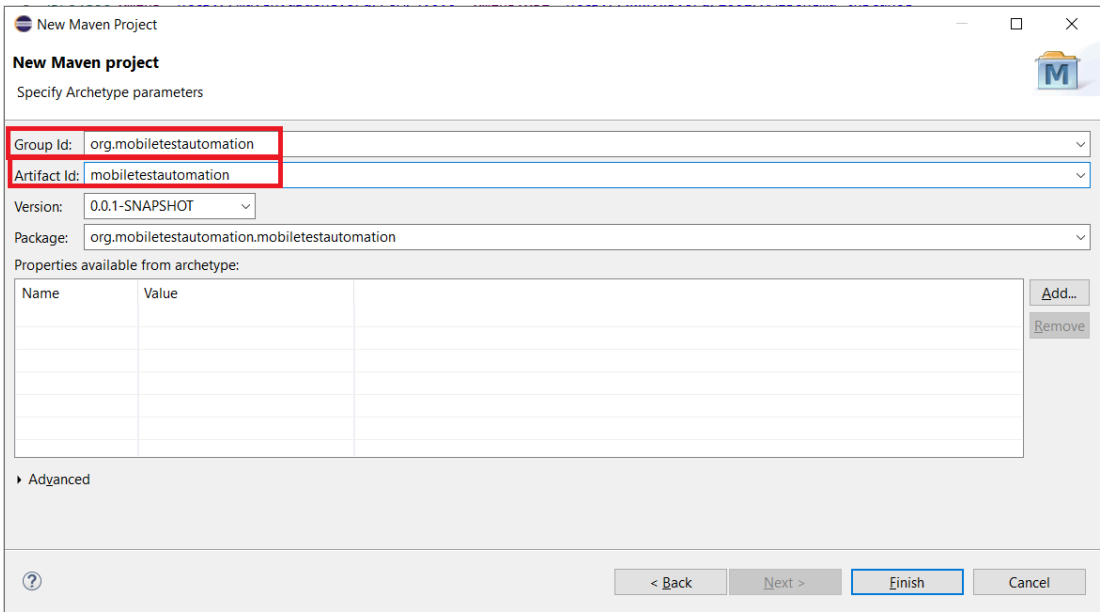


Şekil 3.25: Archetype seçimi

Group Id: Grup kimliği, proje veya organizasyonunun benzersiz bir tanımlayıcısıdır. Genellikle ters alan adı (reverse domain name) şeklinde yazılır. Örneğin, "com.example" gibi. Bu, proje veya organizasyonunuzun adını temsil eder.

Artifact Id: Sanat eseri kimliği, Maven deposunda (repository) proje için benzersiz bir tanımlayıcıdır. Genellikle proje adı veya modül adı olarak kullanılır. Örneğin, "my-project" gibi.

Bu alana bilgileri girip “Finish” butonuna tıklanır.



Şekil 3.26: Proje oluşturmayı tamamlama

3.2.2 Bağımlılıkların Tanımlanması

Proje oluştururken gerekli bağımlılıkları pom.xml dosyasına eklenmelidir. Bu bağımlılıklar, projenin Appium, Selenium ve TestNG gibi kütüphaneleri kullanabilmesi için gereklidir.

```
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.141.59</version>
</dependency>

<dependency>
  <groupId>com.comcast.magic-wand</groupId>
  <artifactId>appium</artifactId>
  <version>4.0.3</version>
</dependency>

<dependency>
  <groupId>org.testng</groupId>
  <artifactId>testng</artifactId>
  <version>7.3.0</version>
  <scope>test</scope>
</dependency>

<dependency>
  <groupId>io.appium</groupId>
  <artifactId>java-client</artifactId>
  <version>7.2.0</version>
</dependency>
```

Şekil 3.27: Bağımlılıkların tanımlanması

3.2.3 Hazırlık İşlemleri

Bu bölümde, uygulama testi için gerekli hazırlık işlemleri yapılır. Bu işlemler arasında, test sürücüsünün başlatılması, test cihazının ve uygulamanın yapılandırılması, sayfa öğelerinin tanımlanması ve bekleme sürelerinin ayarlanması gibi adımlar yer alır. Bu işlemler, testin başarılı bir şekilde yürütülebilmesi için gereklidir ve testin başlamadan önce gerçekleştirilir.

```
public AppiumDriver<WebElement> driver;
public WebDriverWait wait;


By consultantButton = By.id("mobi.appcent.apponte:id/btnInstitutional");

By loginButton = By.id("mobi.appcent.apponte:id/ibSignIn");
By numberText = By.id("mobi.appcent.apponte:id/etPhoneNumber");
By sendCodeButton = By.id("mobi.appcent.apponte:id/btnSendCode");


@BeforeTest
public void beforeTest() {



    try {
        DesiredCapabilities cap;
        cap = new DesiredCapabilities();

        cap.setCapability("deviceName", "Pixel 4 API 30");
        cap.setCapability("platformName", "Android");
        cap.setCapability("udid", "emulator-5554");
        cap.setCapability("platformVersion", "11.0");
        cap.setCapability("appPackage", "mobi.appcent.apponte");
        cap.setCapability("appActivity", "mobi.appcent.apponte.ui.activity.login.LoginActivity");
        cap.setCapability("skipUnlock", true);
        cap.setCapability("noReset", false);


        driver = new AndroidDriver<WebElement>(new URL("http://127.0.0.1:4723/wd/hub"), cap);
        wait = new WebDriverWait(driver, 10);


    } catch (MalformedURLException mx) {
        System.out.println("Hatalı oluşturulmuş URL istisnası");

    }
}
```

Şekil 3.28: Hazırlık işlemleri

3.2.3.1 Test Sınıfının Üye Değişkenlerinin Tanımlanması

Bu kısım, test sınıfının üye değişkenlerini tanımlar. AppiumDriver ve WebDriverWait sınıflarından driver ve wait adında iki üye değişken tanımlanır. Bu değişkenler, test sırasında kullanılacak olan sürücü ve bekleme nesnelerini temsil eder.

```
public AppiumDriver<WebElement> driver;
public WebDriverWait wait;
```

Şekil 3.29: Üye değişkenlerinin tanımlanması

3.2.3.2 Sayfa Öğelerinin Tanımlanması

Bu kısımda, test edilen uygulamanın kullanıcı arayüzündeki bazı öğelerin tanımları bulunur. Her bir öğe, By sınıfının id metodu kullanılarak tanımlanır. Bu öğeler, test sırasında bu sayfa öğelerine erişmek için kullanılacaktır.

```
By consultantButton = By.id("mobi.appcent.apponte:id/btnInstitutional");

By loginButton = By.id("mobi.appcent.apponte:id/ibSignIn");
By numberText = By.id("mobi.appcent.apponte:id/etPhoneNumber");
By sendCodeButton = By.id("mobi.appcent.apponte:id/btnSendCode");
```

Şekil 3.30: Sayfa öğelerinin tanımlanması

3.2.3.3 BeforeTest Anotasyonu

Bu kısım, TestNG testi başlamadan önce çalıştırılacak hazırlık işlemlerinin tanımlandığı metodu belirtir. @BeforeTest anotasyonu, bu metodu testlerin başlamasından önce çağırır.

```
@BeforeTest
public void beforeTest() {
```

Şekil 3.31: BeforeTest anotasyonu

3.2.3.4 DesiredCapabilities Oluşturma ve Ayarlanması

DesiredCapabilities sınıfı kullanılarak cihaz ve uygulama ayarları belirlenir. Bu ayarlar, testin hangi cihazda ve uygulamada çalışacağını tanımlar.

```
DesiredCapabilities cap;
cap = new DesiredCapabilities();

cap.setCapability("deviceName", "Pixel 4 API 30");
cap.setCapability("platformName", "Android");
cap.setCapability("udid", "emulator-5554");
cap.setCapability("platformVersion", "11.0");
cap.setCapability("appPackage", "mobi.appcent.apponte");
cap.setCapability("appActivity", "mobi.appcent.apponte.ui.activity.login.LoginActivity");
cap.setCapability("skipUnlock", true);
cap.setCapability("noReset", false);
```

Şekil 3.32: DesiredCapabilities oluşturma

3.2.3.5 AppiumDriver ve WebDriverWait Oluşturma

Belirlenen ayarlar kullanılarak AppiumDriver ve WebDriverWait oluşturulur. AppiumDriver, testlerin gerçekleştirileceği cihazı ve uygulamayı kontrol etmek için kullanılır. WebDriverWait, belirli koşullar gerçekleşene kadar beklemek için kullanılır.

```
driver = new AndroidDriver<WebElement>(new URL("http://127.0.0.1:4723/wd/hub"), cap);
wait = new WebDriverWait(driver, 10);
```

Şekil 3.33: AppiumDriver ve WebDriverWait oluşturma

3.2.3.6 Hata Yakalama

Try-catch bloğu sayesinde eğer URL oluşturma işlemi sırasında bir hata oluşursa, yakalanır ve bu durumla ilgili bir mesaj yazdırılır.

```
} catch (MalformedURLException mx) {
    System.out.println("Hatalı oluşturulmuş URL istisnası");

}
```

Şekil 3.34: Hata yakalama

3.2.4 Test Senaryosunun Oluşturulması

Bu bölümde uygulamanın giriş işlemlerinin gerçekleştirildiği bir test senaryosunu oluşturuyoruz. Bu işlemler arasında cihaz ve uygulama ayarlarının belirlenmesi, kullanıcı girişinin simüle edilmesi ve OTP (tek kullanımlık şifre) girişinin manuel olarak tamamlanması bulunuyor.

3.2.4.1 Test Anotasyonu

Uygulamanın giriş işlemlerini içeren test metodu bulunmaktadır. Bu metot, @Test annotasyonu ile işaretlenir ve test senaryosunun uygulandığı ana bölümdür.

```
@Test
Run | Debug
public void test() throws InterruptedException {
```

Şekil 3.35: Test Anotasyonu

3.2.4.2 ImplicitWait Kullanımı

Implicit bekleme süresinin ayarlanması sağlanır. Bu süre boyunca, WebDriver belirli bir öğeyi ararken, öğe bulunana kadar belirtilen süre boyunca bekler.

```
driver.manage().timeouts().implicitlyWait(5, TimeUnit.SECONDS);
```

Şekil 3.36: ImplicitWait kullanımı

3.2.4.3 Üyelik Tipi Seçimi

Üyelik tiplerinden ID'si tanımlanmış olan üyelik tipi bulunur ve tıklanır.

```
WebElement consultantSelect = driver.findElement(consultantButton);
consultantSelect.click();
```

Şekil 3.37: Üyelik tipi seçimi

3.2.4.4 Giriş Butonuna Tıklama

Uygulamada “loginButton” öğesi yani giriş butonu aranır ve bulunca giriş butonuna tıklanır.

```
WebElement loginSelect = driver.findElement(loginButton);
loginSelect.click();
```

Şekil 3.38: Giriş butonuna tıklama

3.2.4.5 Telefon Numarası Girişi

Kullanıcı telefon numarasını girmek için “numberText” öğesi aranır ve metin kutusuna numarasını girer.

```
WebElement phoneNumberSelect = driver.findElement(numberText);
driver.getKeyboard().pressKey("55[redacted]");
```

Şekil 3.39: Telefon numarası girişi

3.2.4.6 Kod Gönderme

Doğrulama kodunu göndermek için “sendCodeButton” öğesi aranır gerekli butona tıklanır..

```
WebElement sendCodeSelect = driver.findElement(sendCodeButton);
sendCodeSelect.click();
```

Şekil 3.40: Kod gönderme

3.2.4.7 Manuel OTP Girişi

OTP girişi manuel olarak yapılır ve bu işlem için 10 saniye beklenir.

```
//OTP Girişi manuel olarak yapılıyor
Thread.sleep(10000);
```

Şekil 3.41: Manuel OTP girişi

3.2.5 Son Test Sonrası Temizlik

Testlerin tamamlanmasının ardından uygulamanın ve sürücünün kapatılmasını sağlarız. Bu sayede testlerin sonunda uygulama ve sürücü kaynaklarından düzgün bir şekilde temizlenir ve sistem üzerinde gereksiz yük oluşturulmaz.

```
@AfterTest
public void teardown() {

    driver.quit();

}
```

Şekil 3.42: Son test sonrası temizlik

3.2.5.1 AfterTest Anotasyonu

AfterTest anotasyonu, TestNG tarafından sağlanan bir özelliktir. Bu anotasyon, her test metodunun çalışmasının ardından otomatik olarak çağrılır ve temizlik işlemlerinin gerçekleştirilmesi için kullanılır.

```
@AfterTest
public void teardown() {
```

Şekil 3.43: AfterTest anotasyonu

3.2.5.2 Driver Kapatma

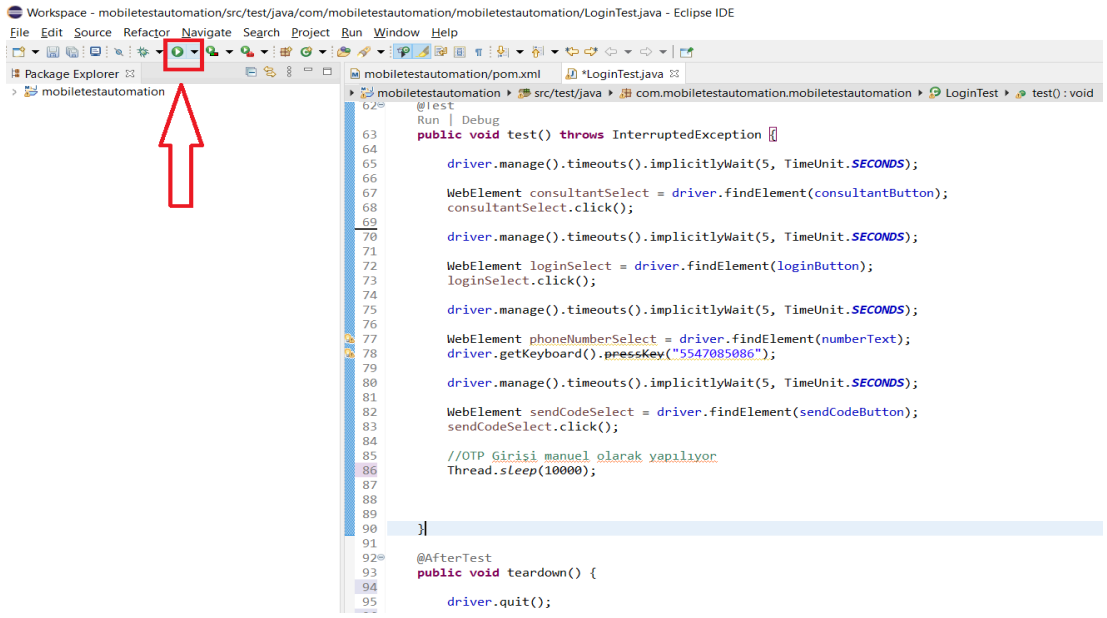
Driver.quit() metodu, WebDriver örneğini kapatır ve kullanılan tarayıcı oturumunu sonlandırır. Bu, testlerin tamamlandığında kaynakların serbest bırakılmasını ve gereksiz bellek tüketimini önlemek için önemlidir.

```
driver.quit();
```

Şekil 3.44: Driver kapatma

3.2.6 Test Senaryosunun Çalıştırılması

Örnek test otomasyonu projemizin tüm kodlarını yazdıktan sonra artık "Run" butonuna tıklayarak test otomasyonunun çalışması sağlanır.



Şekil 3.45: Test otomasyonunun çalıştırılması

3.3 RPA'in Kullanılabilirliği

Günümüzde, işletmeler mobil uygulamaları kullanarak müşterilere daha iyi hizmet sunmayı ve verimliliği arttırmayı hedeflemektedir. Mobil uygulama geliştirme süreci karmaşıktır ve birçok aşamayı içerir, bu nedenle test otomasyonu, mobil uygulamaların doğru ve güvenilir bir şekilde çalışmasını sağlamak için önemlidir. Bu bağlamda, Robotik Süreç Otomasyonu (RPA), mobil uygulama geliştirme ve test otomasyonunda önemli bir rol oynayabilir.

RPA, tekrarlayıcı ve rutin işleri otomatikleştirerek insan kaynaklarını serbest bırakarak iş süreçlerinin verimliliğini arttırmayı amaçlar. Geleneksel olarak, RPA genellikle masaüstü uygulamaları ve web tabanlı uygulamalarla ilişkilendirilmiştir. Ancak, son zamanlarda mobil uygulamaların yaygınlaşmasıyla birlikte, RPA'nın mobil uygulama alanında da kullanılma potansiyeli artmıştır.

Mobil uygulamaların geliştirilmesi ve test edilmesi süreçlerinde RPA'nın kullanılabilirliği, bir dizi avantaj ve zorluklarla birlikte gelir. Öncelikle, RPA, mobil uygulamaların farklı platformlarda ve cihazlarda sorunsuz bir şekilde çalışmasını sağlamak için tekrarlayan görevleri otomatikleştirme yeteneği sunar. Bu, test süreçlerinin hızını artırabilir ve insan hatalarını azaltabilir. Ayrıca, RPA'nın ölçeklenebilirliği, büyük ölçekli test süreçlerini yönetme yeteneği sunar.

Ancak, mobil uygulamaların karmaşıklığı ve sürekli değişen doğası, RPA'nın mobil test otomasyonunda bazı zorluklarla karşılaşmasına neden olabilir. Mobil uygulamalar genellikle dinamik ve etkileşimlidir, bu da test senaryolarının karmaşıklığını artırır. Ayrıca, farklı mobil platformlar ve cihazlar arasındaki uyumluluğun sağlanması gerekliliği, RPA'nın test senaryolarını daha karmaşık hale getirebilir.

3.3.1 Kullanılabileceği Örnekler

RPA'i Mobil uygulama yapımında veya test aşamasında kullanabilmek mümkündür. Bununla ilgili bazı proje önerileri aşağıdadır.

3.3.1.1 Yemek Sipariş Uygulaması

Müşterinin telegram üzerinden bir kanalda "döner" diye aratma yaptığı zaman RPA'in olduğu cihaz farklı yemek uygulamaları üzerinden fiyat ve restoran bilgilerini alarak yine müşteriye telegram üzerinden iletebilir. Bu sayede farklı yemek uygulamalarını tek çatı altında birleştirmek mümkün olur.

3.3.1.2 Mail ile Test Çalıştırma

Oluşturmuş olduğumuz test senaryolarını gelen mailin konusu ile tetiklemek mümkün. Örneğin; mailin konu kısmında login-success yazıyorsa RPA ile Eclipse üzerinden veya Jenkins üzerinden ilgili test senaryosunun çalışması sağlanabilir. Aynı şekilde çıkan sonuçta mail olarak geri iletilebilir.

3.3.1.3 Mail ile Uygulama Yapımı

RPA kullanarak gelen mailin içeriğindeki web sitesi linkini kullanarak web view özellikli basit bir mobil uygulama yapmak gayet mümkündür.

Bu ve benzeri projeler ile RPA'i hem mobil uygulamanın yapım sürecinde hem de test otomasyonu sürecinde kullanabiliriz.

Bölüm 4

Sonuç

Bu projede, mobil uygulamalarda test otomasyonunun önemi ve kullanımı üzerine bir değerlendirme yapılmıştır. Araştırma kapsamında, mevcut literatür incelenerek mobil test otomasyonunun temelleri, teknikleri ve uygulama alanları üzerine bir analiz gerçekleştirilmiştir, mobil test otomasyonunun temel bileşenleri olan araçlar incelenmiştir. Selenium, Appium, TestNG gibi popüler araçlar üzerinde durulmuş ve bunların mobil uygulama test otomasyonu için nasıl kullanılabileceği üzerine ayrıntılı bilgiler sunulmuştur. Ayrıca, RPA'nın (Robotik Süreç Otomasyonu) mobil test otomasyonunda nasıl kullanılabileceği de araştırılmıştır.

Mobil uygulamaların hızla artan popülaritesi ve karmaşıklığı, geliştiricileri ve test uzmanlarını doğru ve güvenilir bir test süreci sağlama konusunda zorlamaktadır. Test otomasyonu, bu zorlukların üstesinden gelmek ve uygulama kalitesini artırmak için etkili bir çözüm sunar. Otomatik test senaryoları, tekrar eden görevleri azaltır, hata ayıklama sürelerini kısaltır ve uygulama yayın süreçlerini hızlandırır.

Bu projenin sonuçlarına göre, mobil uygulamalarda test otomasyonu, geliştirme sürecini iyileştirebilir, test verimliliğini artırabilir ve kullanıcı deneyimini olumlu yönde etkileyebilir. Ancak, doğru araçların seçilmesi, test senaryolarının etkili bir şekilde oluşturulması ve sürekli iyileştirme sürecinin benimsenmesi önemlidir.

RPA'in mobil uygulama ve mobil test otomasyonunda kullanılabilirlik durumu, işletmeler için önemli bir tartışma konusudur. RPA'in sunduğu otomasyon yetenekleri, mobil uygulama geliştirme ve test süreçlerini iyileştirebilir ve verimliliği

artırabilir. Test otomasyonlarını RPA ile daha basit hale getirmek mümkün olabilir. Ancak, RPA'in mobil ortamda karşılaştığı zorluklar da dikkate alınmalıdır.

Gelecekteki çalışmalarda, mobil test otomasyonunun daha da geliştirilmesi ve yaygınlaştırılması için yeni yöntemler ve teknolojilerin araştırılması önerilmektedir. RPA'in mobil uygulama alanında kullanılabilirliğini artırmak için daha fazla araştırma ve geliştirme gerekmektedir. Ayrıca, mobil uygulamalarda test otomasyonunun endüstriyel uygulamaları üzerine daha fazla vaka çalışması yapılması ve gerçek dünya senaryolarının incelenmesi gerekmektedir.

Kaynaklar

1: Büşra Takgil, Resul Kara. “Android Mobil Uygulamalar için Yazılım Testi”. El-Cezerî Fen ve Mühendislik Dergisi, Cilt: 3, No: 2, 2016, s. 324-328.

2: Daniel L Asfaw. Benefits of Automated Testing Over Manual Testing. (2015) ISSN: 2349-7017

3:Güneş Kuday, 2014. “Android Mobil Uygulamalar için Yazılım Testi”, 1 Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Düzce Üniversitesi, Düzce -Türkiye.

4: Keytorc Yazılım Test Hizmetleri. “Mobil Uygulamalar Nasıl Test Edilir?”. https://keytorc.com/blog/mobil-uygulamalar-nasil-test-edilir/.

5: Robusta AI. “Yazılım Test Otomasyonu İçin RPA Nasıl Kullanılır?”. https://robusta.ai/tr/blog/yazilim-test-otomasyonu-icin-rpa-kullanimi/.