SMARTPHONETARAKO APLIKAZIOEN PROGRAMAZIOA TITANIUM ERABILITA

(APUNTEAK)

Edukien aurkibidea

SMARTPHONETARAKO APLIKAZIOEN PROGRAMAZIOA TITANIUM ERABILITA

(APUNTEAK)

1. Deskribapena.

Apunte hauek *Smartphonetarako aplikazioen programazioa Titanium erabilita* ikastaroari begira prestatuak izan dira. Ikastaroa ekainaren 26 eta 27an Markeskua Jauregian, UEUko Eibarko egoitzan emango da.

Apunte hauek osatzeko, batez ere http://www.appcelerator.com web guneko informazioa erabili da. Honetaz gain, Seven days with Titanium eta bere gaztelaniazko bertsioa Siete días con Titanium eskuliburuak oso lagungarriak izan da ikastaro honen egitura sortzeko, baita bertako adibideak eredu gisa hartzeko ere.

2. Edukia

2.1. Titanium Studio

2.1.1 Aurkezpena

Zer da Titanium

Titanium kode irekiko framework-a da, web garapenerako teknologiak erabiliz (Javascript, HTML, CSS), smartphone nahiz tabletetarako aplikazio natiboak sortzea ahalbidetzen duena.

Appcelerator Titanium[1] Appcelerator Inc. enpresak sortua izan zen 2008ko abenduan. iPhone eta Android-etarako garapenerako soporteak 2009ko ekainean integratu zituen eta iPad-erako garapenerako moldaketak, berriz, 2010eko ekainean. Orduantxe ere Blackberry-rako soportea anuntziatu zuen, baina oraindik beta bertsioan dago.

Titanium iOS eta Android sistema eragileak dituzten smartphonetarako aplikazioak sortzeko erabili daiteke. Bestalde, Titanium Mobile web nahiz mahaigaineko aplikazioak garatzeko ere erabiltzen da, nahiz eta ikastaro honetan ez diren funtzionalitate horiei buruzko argibideak emango.

Titanium Linux, Windows edo MacOX sistema eragilea duten ordenagailuetan instalatu daiteke, dena den, ikastaro honetan Ubuntu 12.04 sistema eragilea duten ordenagailuetan instalatzeko argibideak emango ditugu. iOS sistema eragilea duten smartphonetarako aplikazioak egiteko, beharrezkoa izango da MacOX sistema eragilea edukitzea.

1 Nahiz eta Appcelerator Titanium izan, hemendik aurrera Titanium deituko diogu apunte hauetan

Apache 2.0 lizentziapean erabili daiteke. Ikastaro hau osatzeko informazioa http://www.appcerator.com helbidetik hartu da. Gidak, bideoak, API dokumentazioa, adibideak eta hainbat irudi bertatik hartu dira.

Ezaugarriak

Titanium plataformaren bidez, ondorengo ezaugarriak bere baitan dituzten smartphonetarako aplikazioak sor ditzakegu.

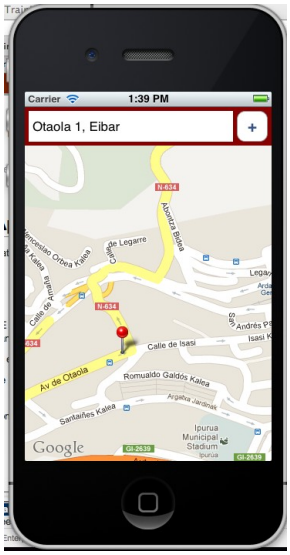
- **Multimedia**

  Irudiak, bideoak eta soinuak modu erraz batean txertatu daitezke aplikazioan.

Iturria: http://www.learningtitanium.com

- **Geolokalizazioa eta mapak**

  Mugikorra daraman pertsonaren kokapena zein den jakin daiteke eta horren ondorioz mapan kokatu, gertuen dituen lekuak, etab. proposatu.



- **Sare sozialak**

Twitter, Facebook bezalako sare sozialetara konektatu gaitezke, bertako informazioa, irudiak aplikazioetan txertatzeko.

- **Datu-baseak**

Datu-baseetara konexioak erraz egin daitezke, modu horretan hiztegiak, etab. sortu ahal izateko.

- **Urruneko datuen atzipena**

Urruneko saretara konektatu gaitezke APIak erabiliz eta bertako informazioa erakutsiz (sare sozialak, RSS irakurgailuak... eginez)

- **Titanium + Plus**

Titanium hedatu daiteke. Beharrezkoa dugun funtzionalitate bat garatua ez badago, moduluak sor ditzakegu eta Titanium-en APIan txertatu. Esaterako, momentu honetan Titanium-ek ez du eskaintzen zuzenketa ortografikoa egiteko modurik.

Lengoaia natiboak

Android sistema eragilea duten smartphonentzat aplikazioak garatzeko Java lengoaia erabiltzen da eta iOS sistema eragilea dutenentzat, berriz, Objective-C.
Hauexek lirateke, gutxienez beharrezkoak liratekeenak aplikazio bat lengoaia natiboan garatzeko:

- Android sistema eragilea duten smartphonentzat aplikazioa lengoaia natiboan egiteko:
  - Java lengoaiaren ezagutza
  - Android SDK (http://developer.android.com/sdk/index.html)
  - Testu editore bat. Gomendagarria, dena den, Eclipse bezalako tresna bat erabiltzea
  - Apache Ant (http://ant.apache.org/bindownload.cgi)

Informazio gehiago: **https://wiki.appcelerator.org/display/td/210+Native+Android+Development**

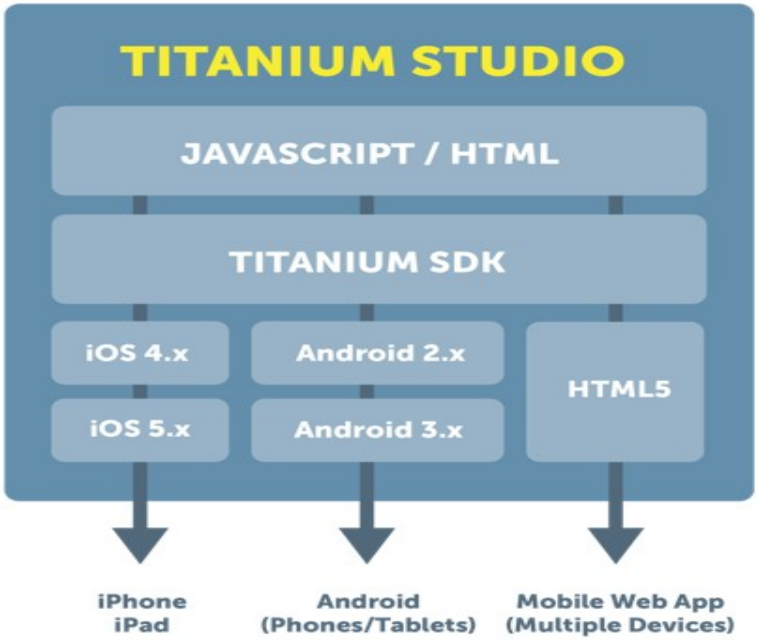
- iOS sistema eragilea duten smartphonentzat aplikazioa lengoaia natiboan egiteko:
  - Ojective-C lengoaiaren ezagutza
  - iOS SDK (http://developer.apple.com)
  - XCode (4 bertsioa, momentu honetan)

Informazio gehiago: **https://wiki.appcelerator.org/display/td/210+Native+iPhone+Development**

Zergatik Titanium

Web garapenerako teknologiak erabiliz iPhone/iPad eta Android-erako aplikazioak erraz sor daitezke. Android-erako aplikazioak sortzeko lengoaia natiboa Java da, eta iOS sistema eragilerako sortzeko, berriz, Objective-C. Titanium erabiliz, web garapenerako tresnak erabiliz eta behin kodetuz, bi sistema erabiletarako aplikazioak sortu ahal izango ditugu.

Ondorengo irudian ikusi ahal den bezala[2], Titanium Mobile erabiliz, aplikazioa Javascript erabiliz idatzi daiteke, Titanium APIari deiak eginez, botoiak, leihoak, kamara... etab.-en funtzionalitateak aplikazioari integratzeko. Titanium zubiak (zeinari Kroll deitzen zaion) kode hau sistema eragile bakoitzari dagokion kode natibora itzuliko du.

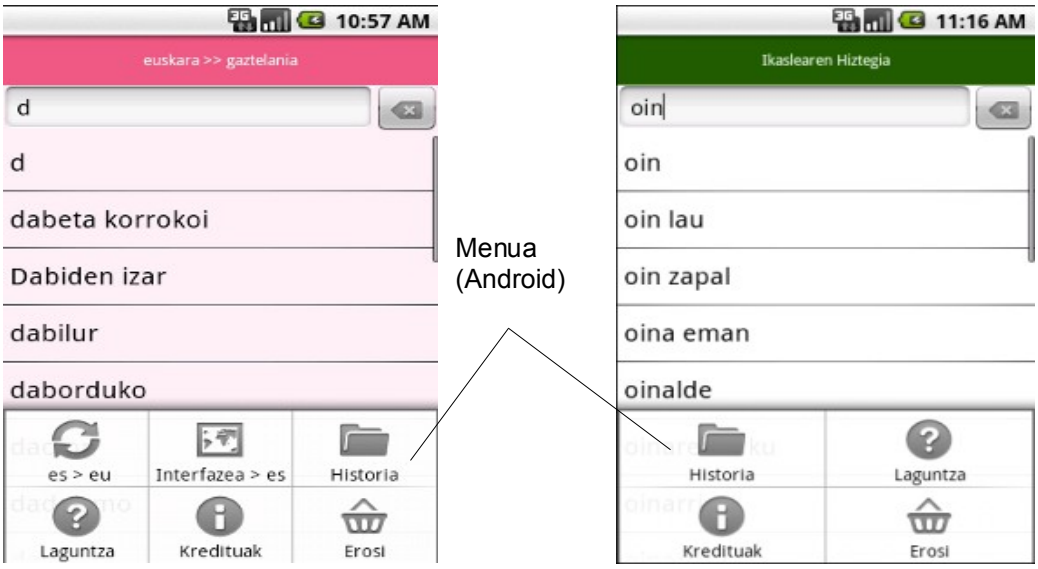


Dena den, badira ezaugarri berezi batzuk gailuaren sistema eragileari lotuta daudenak, eta sistema eragile bakoitzarentzat moldaketak egin behar dira. Beraz, behin kodetu eta ondoren moldatu. Esaterako, iOS sistema eragilea duten aplikazioetan nabigazio sistema berezi bat erabiltzen da Android sistema eragilean existitzen ez dena. Hori dela eta, iOSek eskaintzen duen aukera hau aprobetxatu egin behar da, eta horrek desberdin kodetzera behartzen gaitu bi sistemetarako.

2 Irudia Appcelerator guneko dokumentaziotik hartua izan da, dena den zertxobait zaharkitua dago, dagoeneko Android 4.x SDK dagoelako eta iOS 6.x beta ere existitzen da, probak egiten hasteko

Nabigazioa (iOS)

Beste aldetik, esaterako, Android sistema duten mugikorrek badute ezaugarri bat iOS sistemetan existitzen ez dena, menu botoia. Hau sakatzean menu bat irteten da aplikazioaren funtzionalitate ezberdinekin.



Menua (Android)

Bestalde, itxurari dagokionean ere, nahiz eta kontrol batzuk berdin programatu, itxura desberdina hartzen dute pantailetan. Esaterako, Android sistema eragileetan fitxak goian agertzen dira defektuz eta iPhonetan, aldiz, beheran.

Adibidez:

Android	iPhone
	
	
http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Creating_Your_First_Titanium_Apps-section-29004841_CreatingYourFirstTitaniumApps-AndroidandiOSUIComparison	

Titanium erabiltzearen abantailak eta desabantailak

Abantailak	Desabantailak
• Oinarrizko kode berdina erabiliz, iOS, Android eta mobile web aplikazioak sortzeko aukera • Javascript erabiliz ez da beharrezkoa Java edo Objective-C ezagutzarik • Merkatura lehenago iritsi Objective-C edo Java erabiliz baino • APIa eraibiliz mugikorraren hainbat funtzioetara irits gaitezke • Titanium-ek kodea konpilatzen du aplikazioak saltzeko dendetan arazorik ez izateko moduan	• Aplikazioek zertxobait gehiago pisatzen dute eta konpilatutako kodea ez da txukuna • Aplikazioen errendimendua mantsoagoa izan daiteke kasu batzuetan • Android/IOS SDK berri bat kaleratzen denean, Titanium-en integratzeak bere kostua du denboran

Titanium-ekin garatutako aplikazio batzuk

NBC Live

Telebista ikusteaz gain zuzenean parte hartzeko aplikazioa. iOSerako bakarrik, baina Titanium Studiorekin garatua.

Aplikazio gehiago: http://www.builtwithtitanium.com/

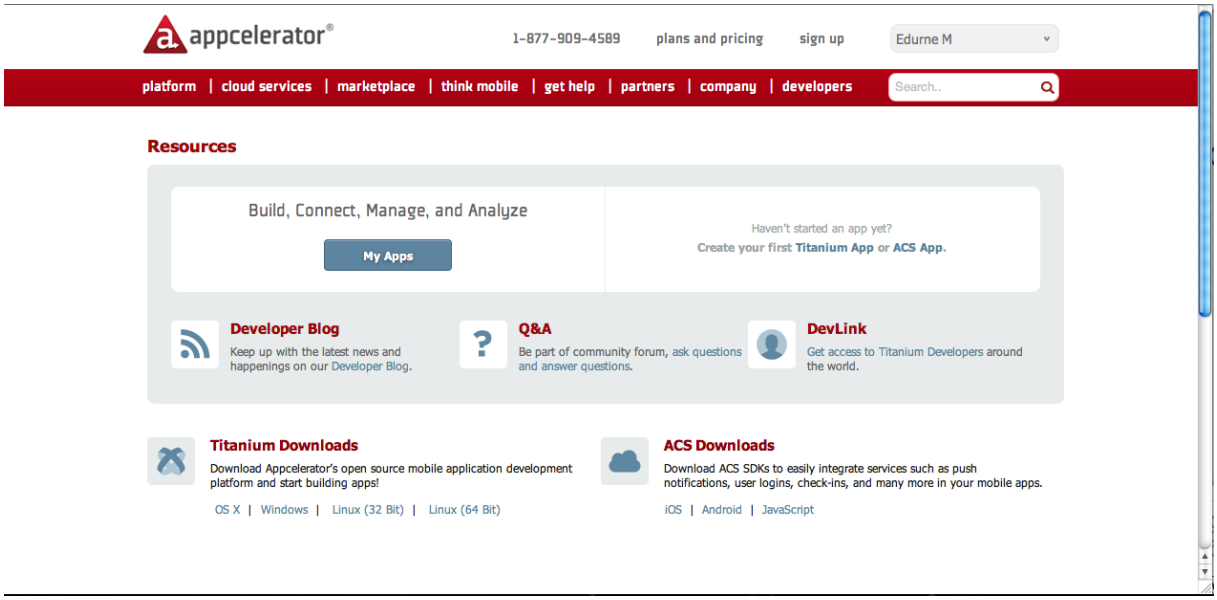
2.1.2 Dokumentazioa

Informazio ugari dago Appcelerator enpresaren gunean, atal desberdinetan:

- **Garatzailearen gunea (**http://my.appcelerator.com**)**

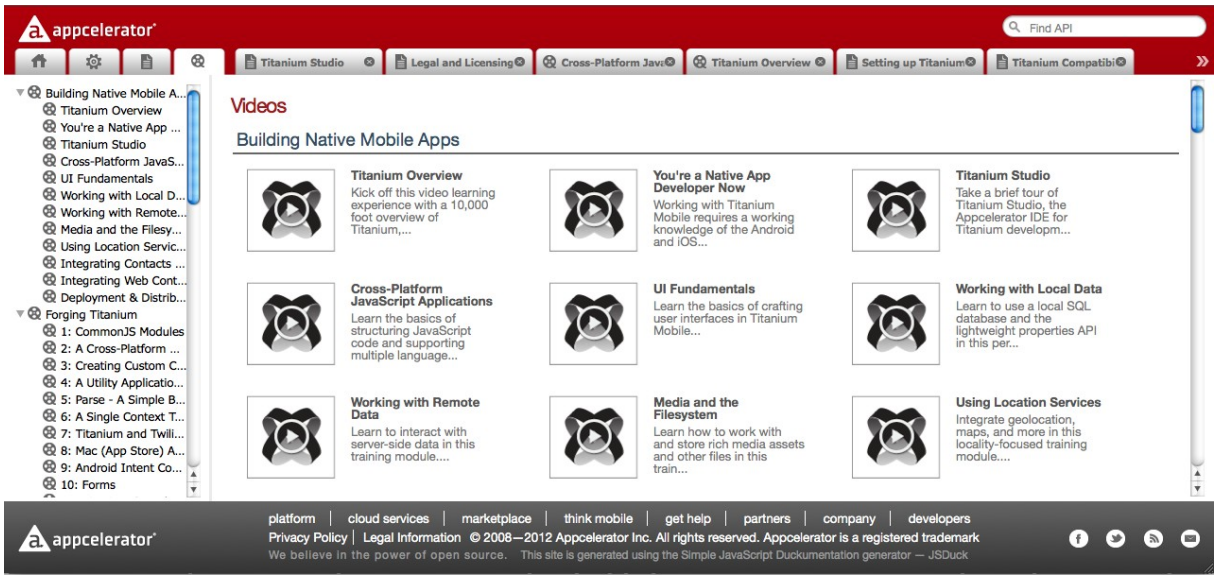
Lehenengo *Appcelerator Network* kontu batean erregistratu behar da eta ondoren informazio-iturri desberdinetarako loturak daude (Titanium jaisteko estekak, Titanium SDK berriei buruzko informazioa, Tutorialak...), baita norberak sortutako aplikazioei buruzko informazioa ere.

Erregistratzeko: http://my.appcelerator.com/auth/signup



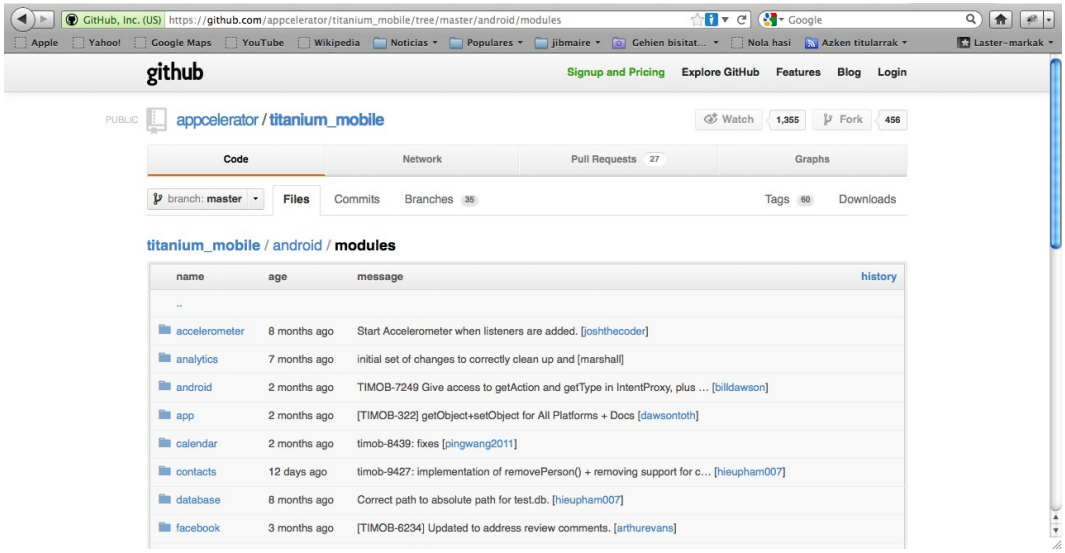
- **Dokumentazioa (http://docs.appcelerator.com)**

Gune honetan, Titanium SDKri buruzko informazioa (bertsioei buruzko informazioa), instalazio gidak, APIaren kontsulta egiteko aukera eta bideo desberdinak topatuko ditugu.



- **Errepositorioa (http://github.com/appcelerator)**

Adibideak, modulu ezberdinen kodeak etab. topatzeko gunea.

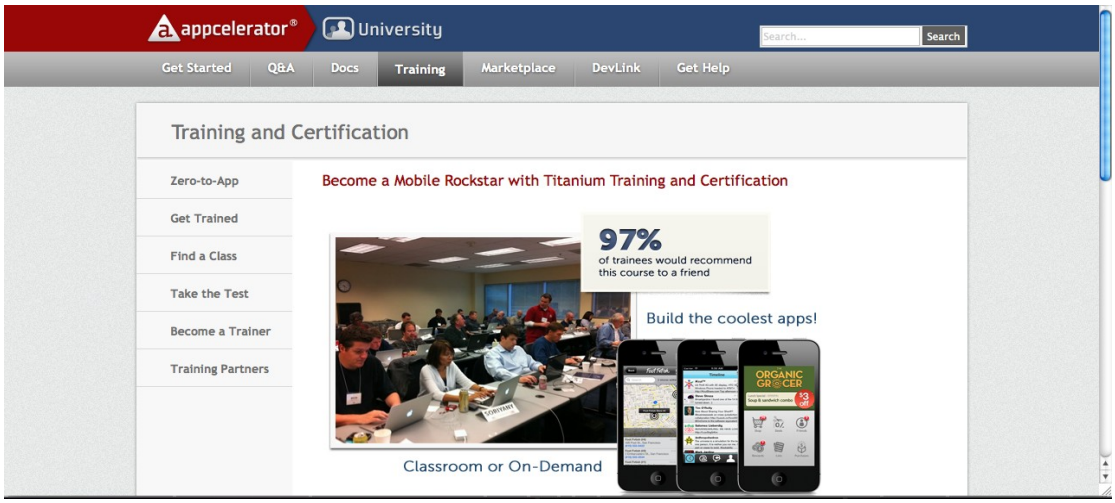


- **Trebakuntza/Zertifikatuak** (http://training.appcelerator.com)

Titanium erabiltzen duen garatzaileak bi zertifikatu lortzeko aukera izango du. Batetik, TCAD (Titanium Certified App Developer) izenekoa eta bestetik TCMD (Titanium Certified Mobile Developer) izenekoa. Bigarrena aurreratuagoentzat da eta lehengoa eskuartean izan behar du garatzaileak, bigarren hori lortu ahal izateko.

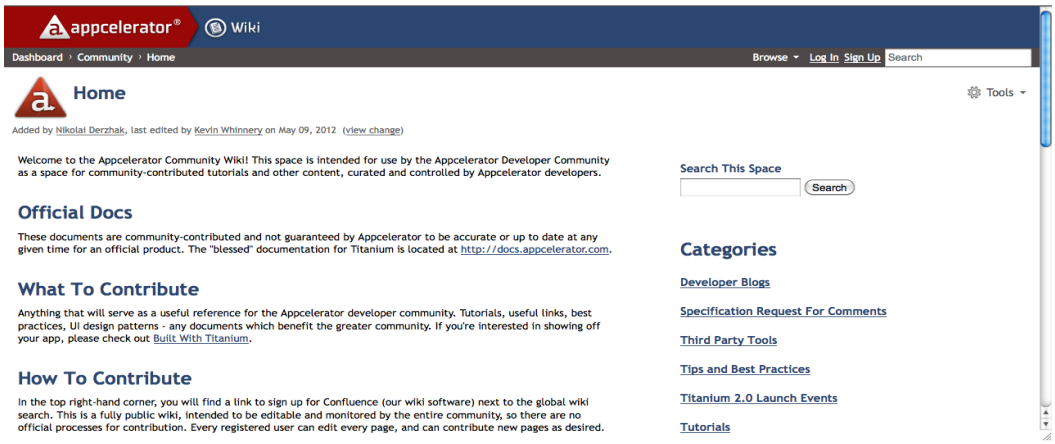
Zertifikatua eskuartean izateak, Appcelerator-en beta programetan parte hartzea, deskontuak Titanium Developer soporte eskaintzetan... izatea suposatzen du.

Gune honetan zertifikatuak lortu ahal izateko informazioa eta laguntza eskaintzen da (laborategiak).



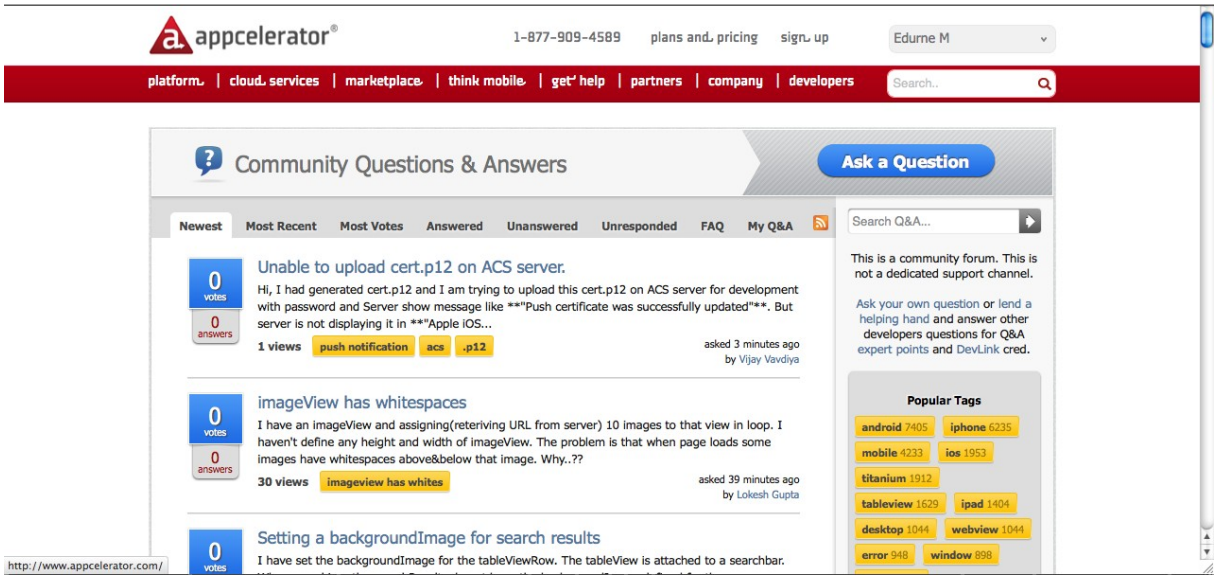
- **Wiki-a** (http://wiki.appcelerator.org)

Appcelerator komunitatearen Wikia, tutorialak, blogak eta beste hainbat informazio partekatzeko erabiltzen da.



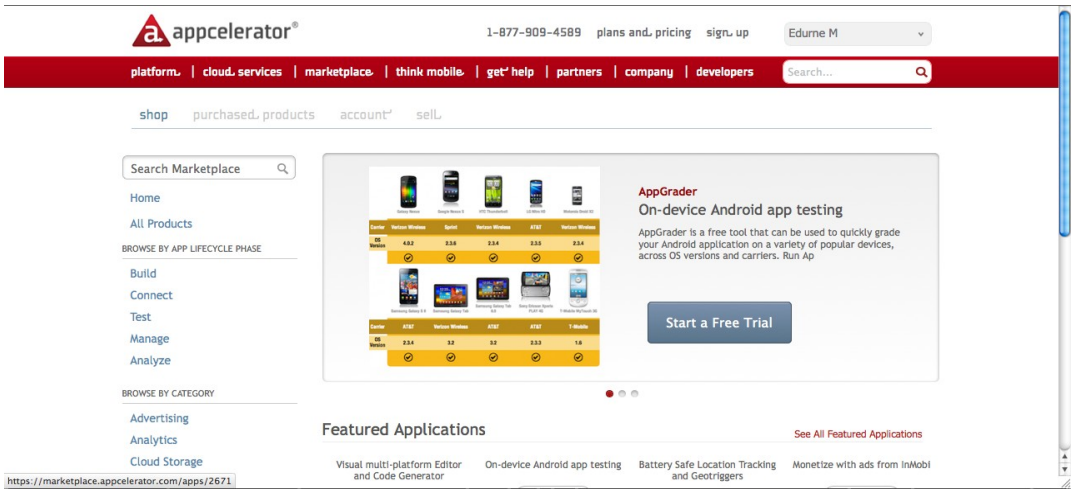
- Galde-erantzunak (http://developer.appcelerator.com/questions)

Garatzaileak, Titaniumekin duen zalantza batekin bertara idatz dezake. Galdera idazteko, beharrezkoa izango da arau batzuk kontutan hartzea.



- Market (http://marketplace.appcelerator.com)

Aplikazioak, moduluak lortzeko gunea. Norberaren produktuak ere bertan jar daitezke.



2.1.3 Instalazioa

Sistema eskakizunak

Titanium Studio instalatu ahal izateko beharrezkoa da:

Sistema eragilea: Windows 7, XP edo Vista, Mac OS X edo Ubuntu

Memoria: 2GB RAM

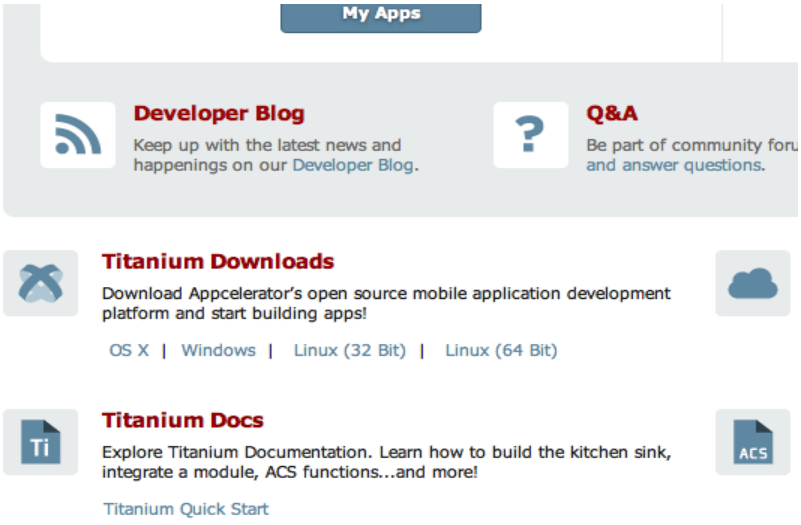
Java Runtime: Oracle JDK

iOS sistema eragilerako aplikazioak garatzeko, beharrezkoa da Mac OS X sistema eragilea erabiltzea eta aplikazioa iPhone batean martxan jartzeko, *Apple Developer Program*an izena ematea beharrezkoa da.

Titanium Studio jaitsi

Dagoeneko prest gaude Titanium Studio instalatzeko, Titanium aplikazioak sortu, kudeatu, garatu, araztu eta hedatzeko ingurunea. Titanium jaitsi eta erabiltzeko, beharrezkoa da *Appcelerator Network* kontu bat izatea. Hementxe sor dezakezu: http://my.appcelerator.com/auth/signup (ikusi 2.1.2 Dokumentazioa atala)

Ondoren, gure sistema eragilearekin bat datorren Titanium-en bertsioa jaitsiko dugu http://my.appcelerator.com gunetik:



Titanium Studio martxan jartzen

Lehenengo aldiz martxan jartzen dugunean *Appcelerator Network* kontuaren erabiltzaile/pasahitzak sartu beharko ditugu eta workspace karpeta aukeratu beharko dugu. Karpeta hau erabiliko du ondoren proiektu fitxategi eta inguruneaen inguruko lehentasun datuak gordetzeko.

Lehenengo aldiz martxan jartzen dugunean ere, automatikoki azken Titanium SDK instalatzen ahaleginduko da. Behin prozesu hau amaituta, komeni da konfirmatzea ez dagoela egin gabeko eguneraketarik, modu honetan:

– *Help -> Check for Updates*: Titanium Studio-ren azken bertsioa instalatuta dagoela bermatzeko

– *Help-> Check for Titanium SDK Updates:* Titanium SDK ofizial guztiak instalatuta daudela bermatzeko

Errepikatu bi pausuak eguneraketarik ez dagoela ziurtatu arte.

Titanium Studio konfiguratzen

Appcelerator enpresaren gunean dagoen *Quick Start* gida honetan oso ondo azaltzen du interfaze bidez nola konfiguratu, eta edozein sistema eragiletan. Bertako pausuak jarraituz erraz egin daiteke instalazioa. Dena den, ikastaro hau emango den laborategian dauden ordenagailuen sistema eragileak Ubuntu 12.04 direla jakinik, sistema eragile horri dagozkion pausu zehatzak azalduko ditugu:

1) Java JDK instalatu

```
sudo apt-get install openjdk-6-jdk
```

2) Git instalatu

```
sudo apt-get install git-core
```

3) Android SDK instalatu eta konfiguratu

a) Jaitsi hemendik: http://developer.android.com/sdk/index.html

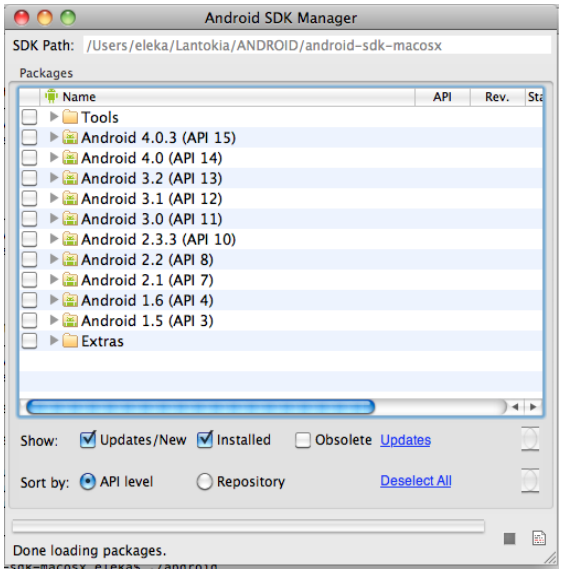
b) Mugitu nahi duzun karpetara:

```
cp home/UEU/Deskargak/android-sdk_r18-linux.tgz /home/UEU/APLIKAZIOAK/
cd /home/UEU/APLIKAZIOAK
```

d) Martxan jarri

```
tar xzvf android-sdk_r18-linux.tgz
PATH=$PATH:/home/UEU/APLIKAZIOAK/android-sdk-linux/tools
export PATH
android
```

e) Jaitsi eta instalatu gutxienez: *Tools* eta *Android 2.2 (API 8)*



Informazio gehiago (Windows/Linux/Mac instalazio gidak):

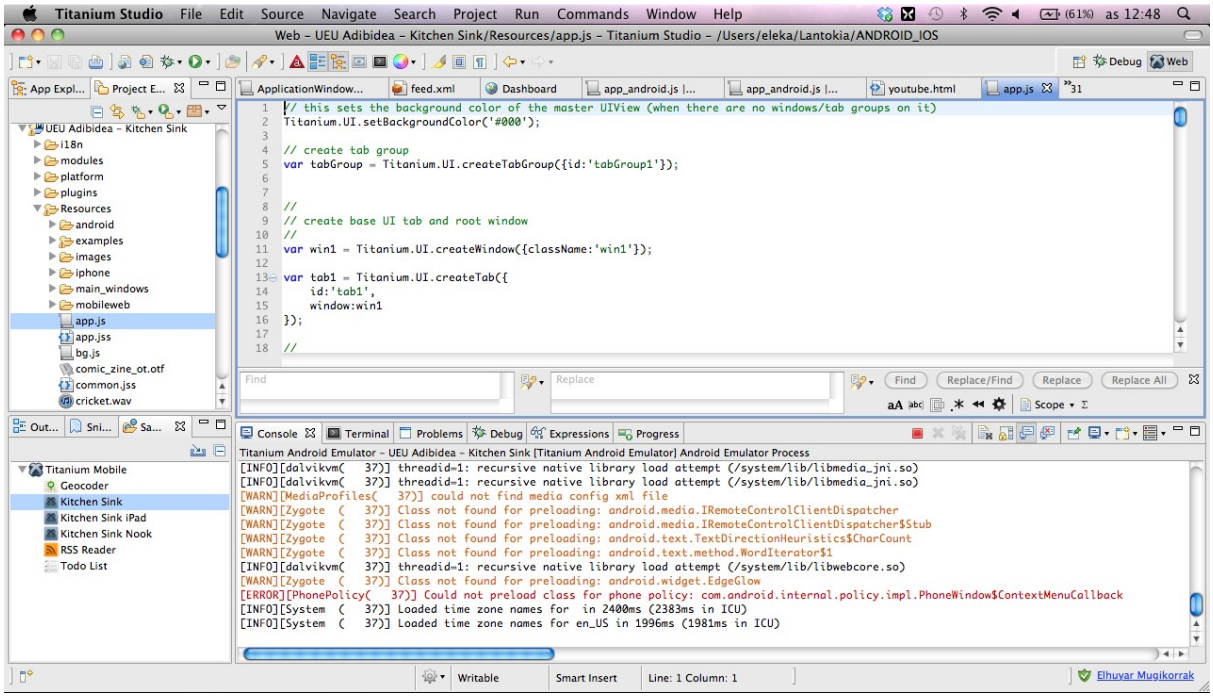
http://developer.appcelerator.com/blog/2010/08/introducing-new-getting-started-guides.html

2.1.4 Ingurunea

Titanium Studio Appcelerator enpresaren doako IDEa (integrated development environment) da. Titanium Studio gure Titanium Mobile aplikazioak idatzi, testeatu eta arazteko erabil daiteke. Gainera, adibideak eta txantiloiak integraturik ditu norbera bere kabuz aplikazioak sortzen errazago has dadin. Titanium plataformak ere Titanium SDK eguneraketak, analitikak eta moduluen erabilera kudeatzen laguntzen du.

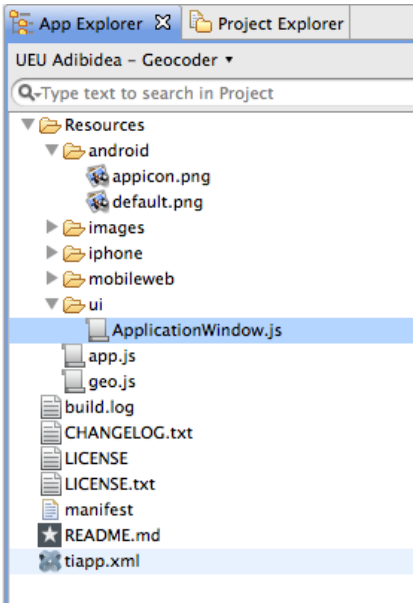
Persperktibak/Multzoak/Bistak

Titanium Studio lehen aldiz martxan jartzen dugunean, defektuz Web perspektiba kargatzen da eta bertan bista ezberdinak lau multzotan *(Tab Group)* banatzen dira:

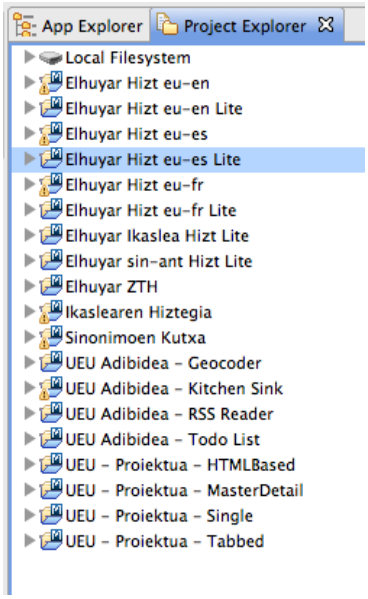


Apunte hauetan Web perspektibari buruz soilik hitz egingo dugu, eta hauexek dira bertan defektuz agertzen diren bista horietako batzuk:

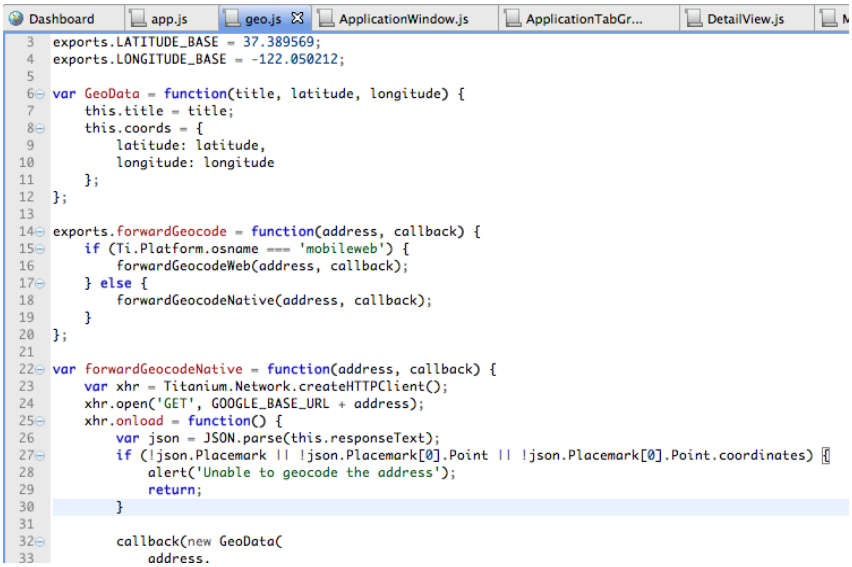
App Explorer: Proiektu zehatz batekin lanean aritzeko.



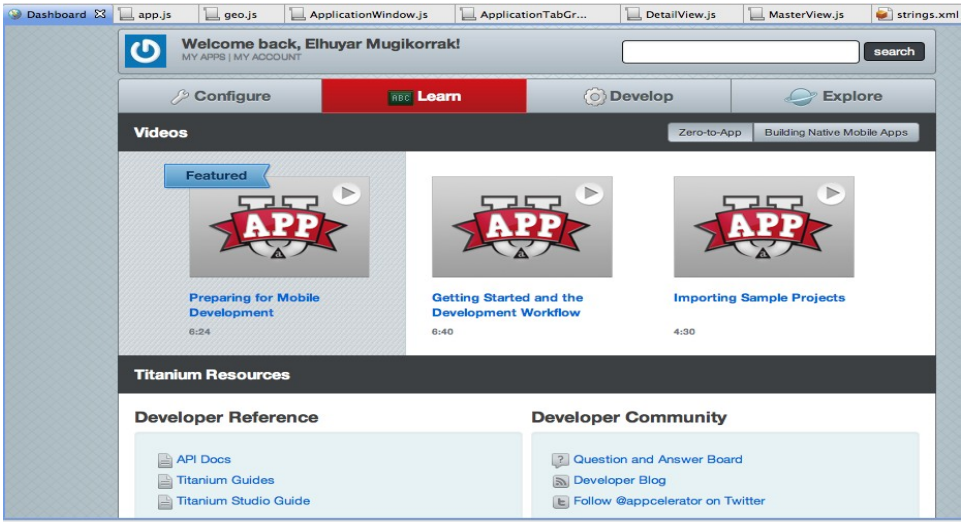
Project Explorer: Proiektu guztien zerrenda erakusten du.



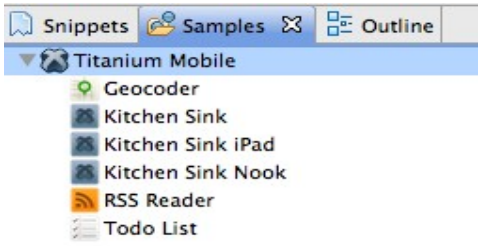
Editor: Proiektuko fitxategi ezberdinak editatzeko (bista berezia da)



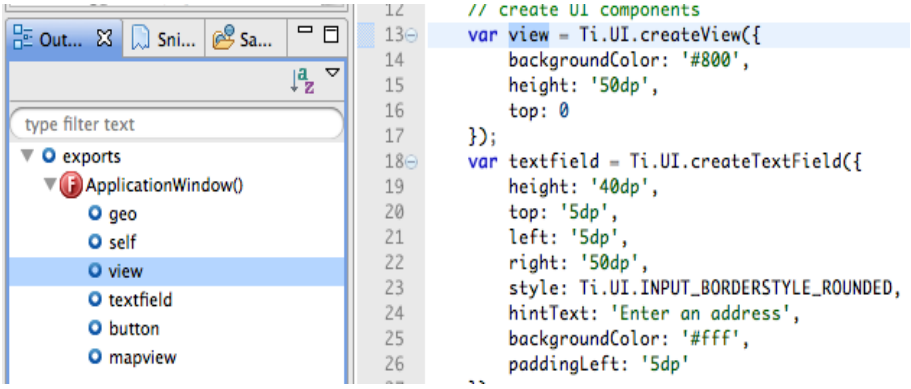
Bista honetan Dashboard izeneko leiho berezi bat irekitzen da defektuz. Bertako estekak erabiliz funtzionalitate interesgarri askotara zuzenean hel gaitezke (leihoa irekita ez badago, goiko menu barran ikonoan klik eginez irekitzen da):



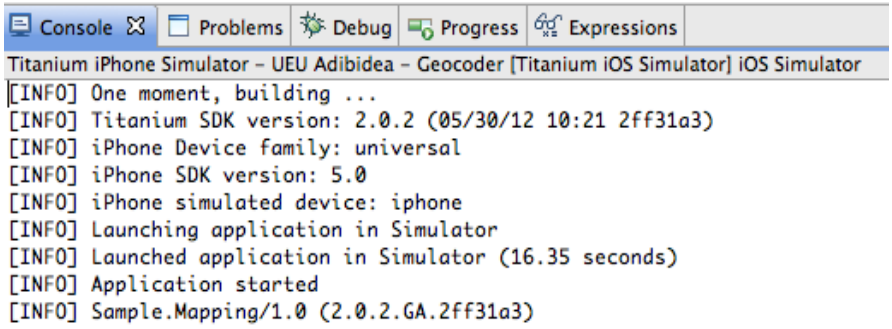
Samples: Titanium Mobilekin lanean hasteko hainbat adibide inportatzeko aukera ematen du.



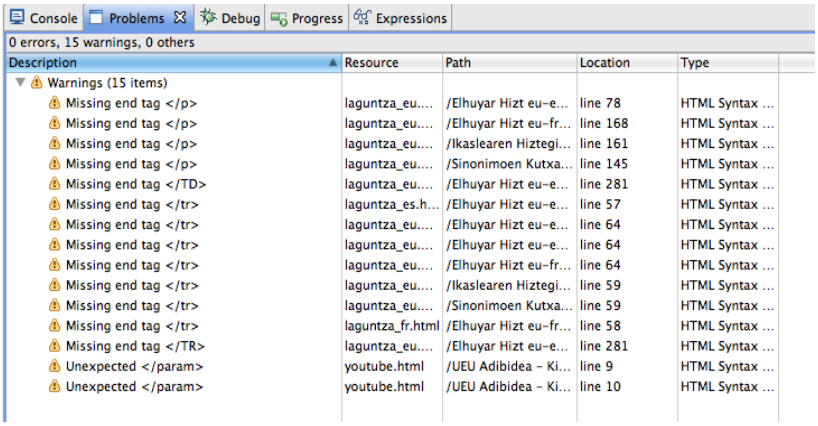
Outline: Kodean agertzen diren aldagai eta funtzioak modu mailakatuan erakusten ditu. Edozeinetan klik eginez bertara eramango gaitu.



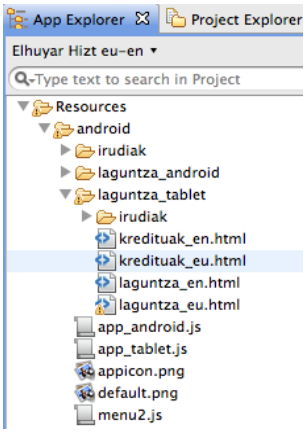
Console: Aplikazioa emuladorean edo gailuan martxan jartzen dugunean, egoera zein den azaltzen da kontsolan. Kodean zehar ere trazak jarri daitezke (esaterako, *info, debug* motakoak) eta bertan agertuko dira ere.



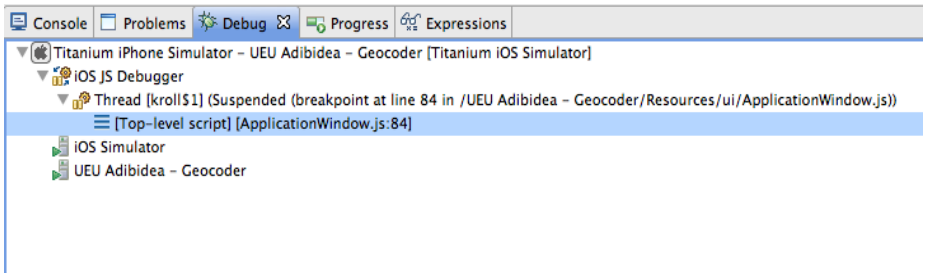
Problems: Aplikazioan ditugun arazoen berri ematen digu bista honek. Abisuak edo erroreak izan daitezke.



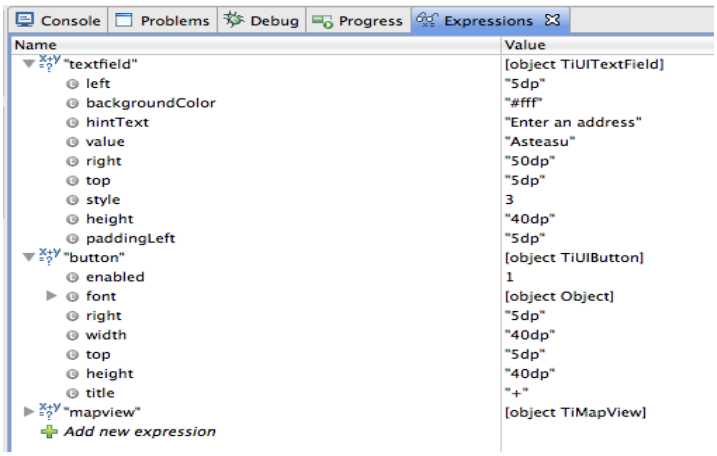
Aplikazioaren batean errore edo abisuren bat konpontzeke egonez gero, *App Explorer* edo *Project Explorer* bistetan ikono baten bidez markatuko dira fitxategia eta gaineko karpetak ere.



Debug: Aplikazioa arazterako garaian, bertan adieraziko zaigu aplikazioaren zein puntutan lanean ari garen.



Expressions: Arazten ari garen bitartean, bertan ikus ditzakegu aldagai ezberdinen balioak.



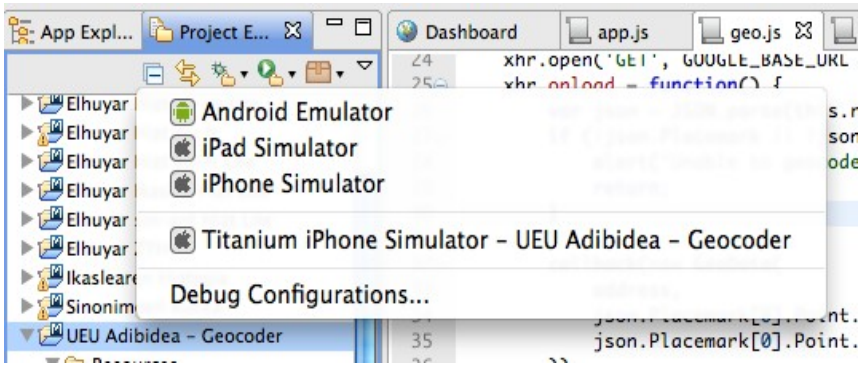
Ohar batzuk:

- Edozein bista handitu nahi izanez gero, fitxaren tituluaren gainean klik egin eta pantaila osoa hartzen du
- Bistak multzo batetik bestera mugitu daitezke arrastatuz (*Editor* bistakoak izan ezik).

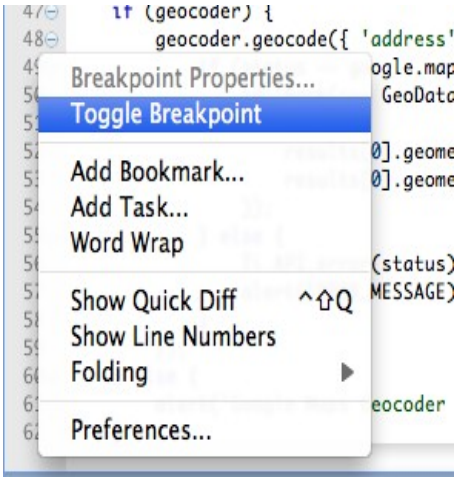
Debug/Run/Publish

Prozesu hauek *App Explorer* edo *Project Explorer* bistetatik martxan jar daitezke:

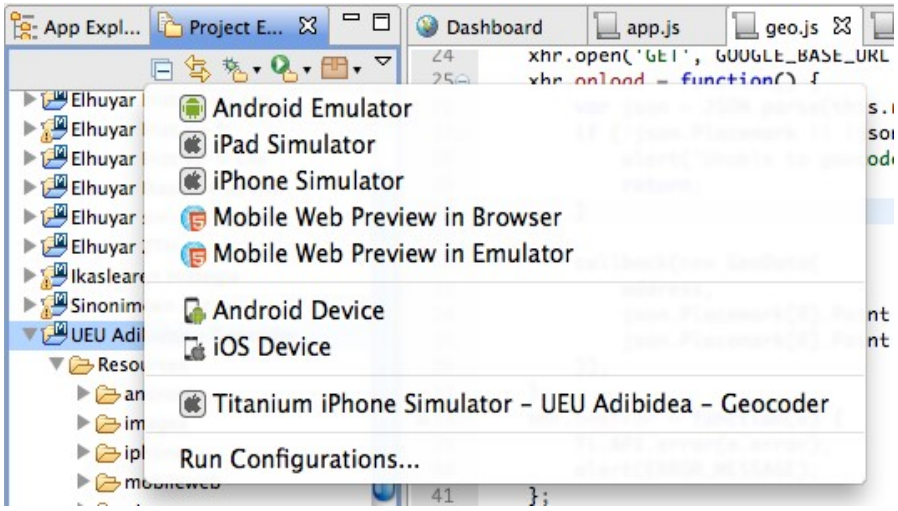
Debug : Aplikazioa martxan jartzen du Android/iPhone/iPad simuladorean eta arazteko aukera eskaintzen du.



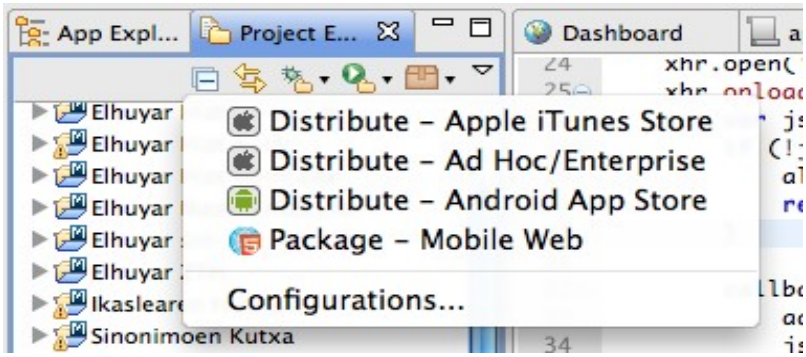
Lerro konkretu bateko informazioa arazteko beharrezkoa izango da eten-puntuak jartzea. Horretarako, kode lerroaren gainean klik egin eta *Toggle Breakpoint* sakatu behar da. Ondorioz, lerroa markatuta geratuko da borobil urdin baten bitartez.



Run: Aplikazioa martxan jartzen du Android/iPhone/iPad simuladorean edo Android/iOS gailuetan (zuzenean)

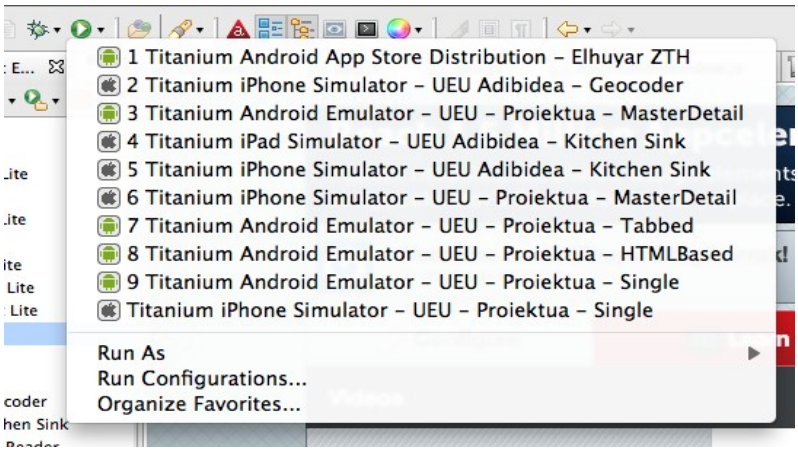


Publish: Aplikazioak *Google Play* edo *App Store*n banatuak izan daitezen prozesatzen ditu.



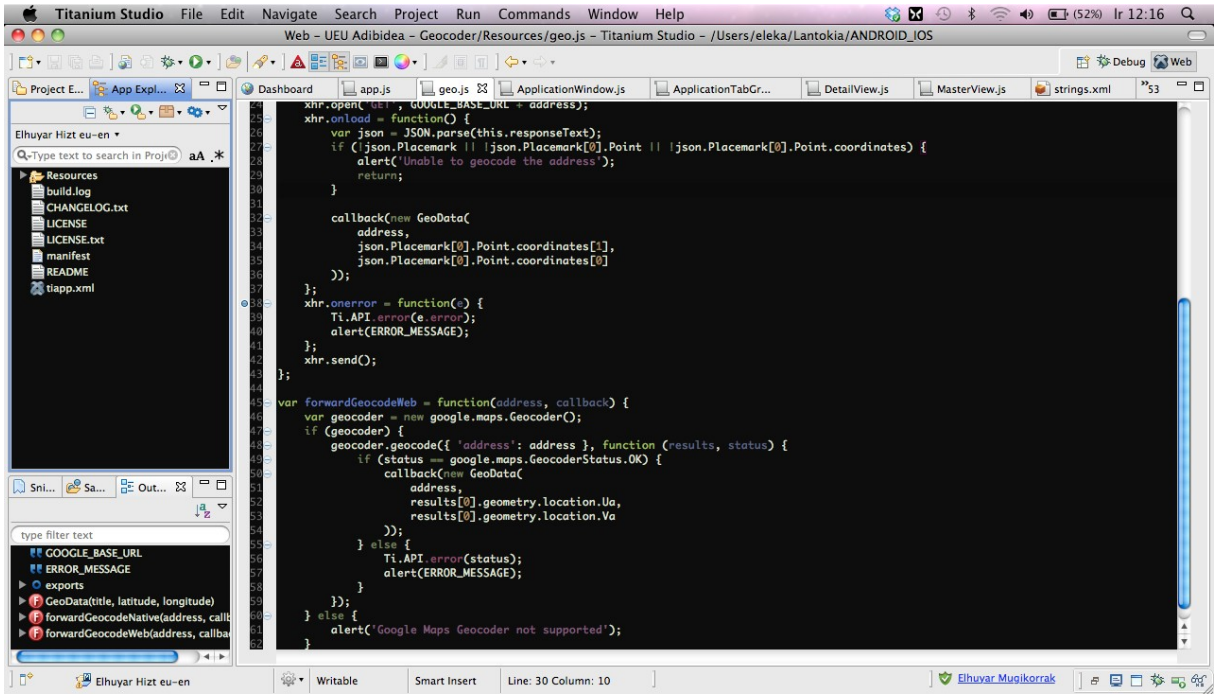
Oharrak:

Run/Debug/Publish funtzionalitateak goiko menutik ere exekutatu daitezke. Debug egiteko,  ikonoa erabiliz eta *Run/Publish* egiteko  ikonoa erabiliz. Klik eginez gero, azkena burutu den prozesua errepikatuko da, baina ondoko gezian klik eginez gero, zerrenda bat agertuko zaigu aurrez egindako beste prozesu bat martxan jarri nahiko bagenu.



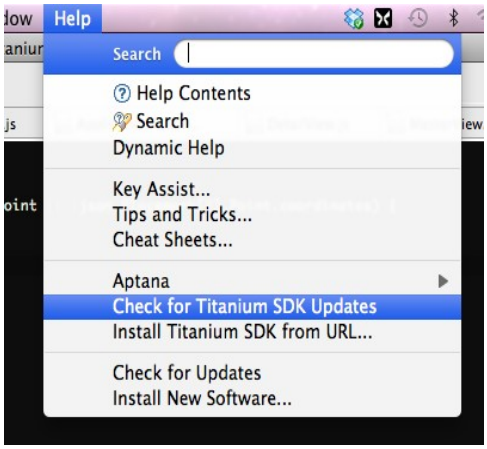
Pertsonalizatu

Ingurunea pertsonalizatu daiteke koloreak aldatuz, makroak gehituz, tekla-lasterbideak aldatuz, lerro zenbakiak ezkutatuz etab.



Titanium Studio / SDK eguneraketak

Komeni da tarteka Titanium Studio eta SDKaren eguneraketak burutzea eta azken bertsioak martxan edukitzea.



Informazio gehiago:

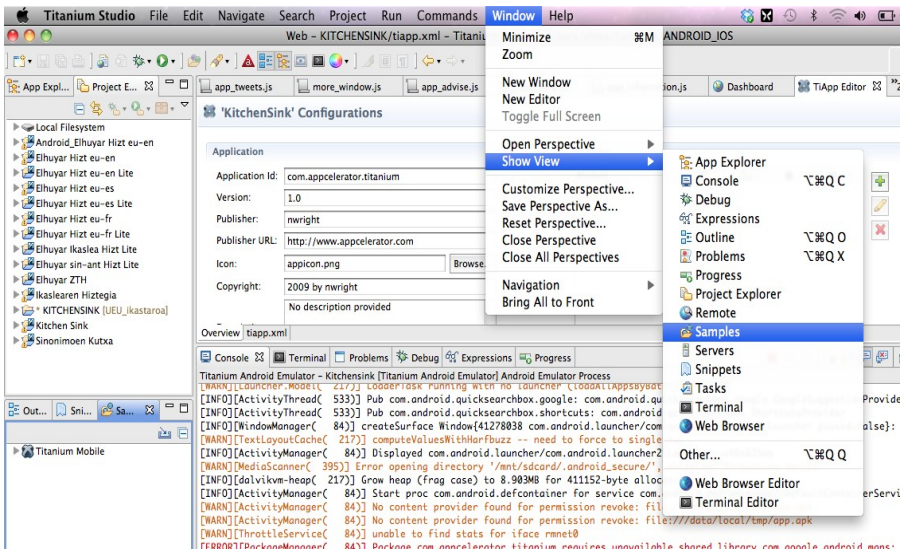
http://docs.appcelerator.com/titanium/2.0/index.html#!/video/28824081

http://docs.appcelerator.com/titanium/2.0/index.html#!/video/28822084 (zatitxo bat dago)

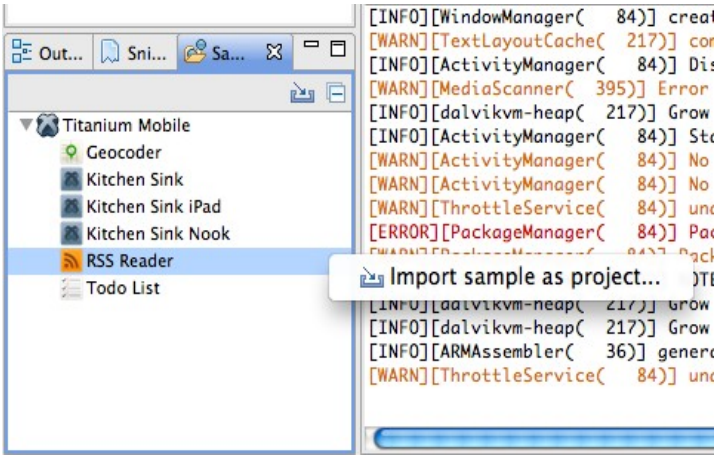
2.1.5 Adibideak inportatzen

Adibideak zuzenean inportatu daitezke Titanium Studio erabiliz. Horretarako, ondorengo pausuak jarraitu behar dira:

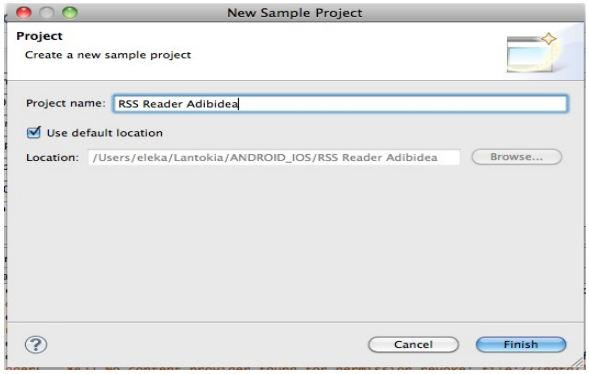
1. Samples bista ireki behar da *(Window -> Show View -> Samples)*



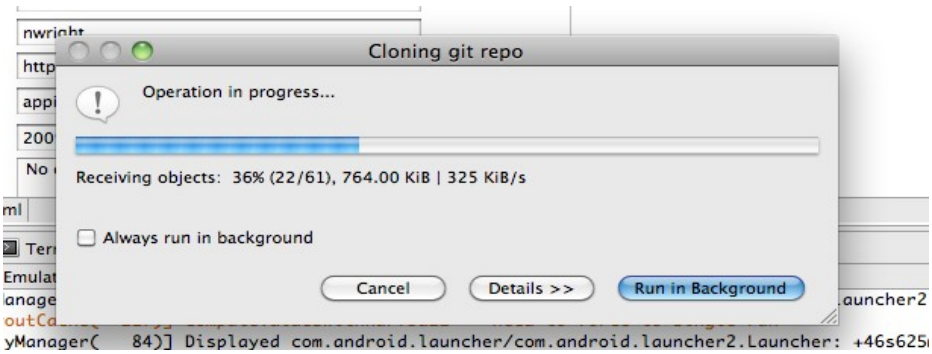
2. Behin bista irekita dugula, hautatu zerrendatik inportatu nahi duzun adibidea, eskuin botoiarekin klik egin eta aukeratu *Import sample as project...*:



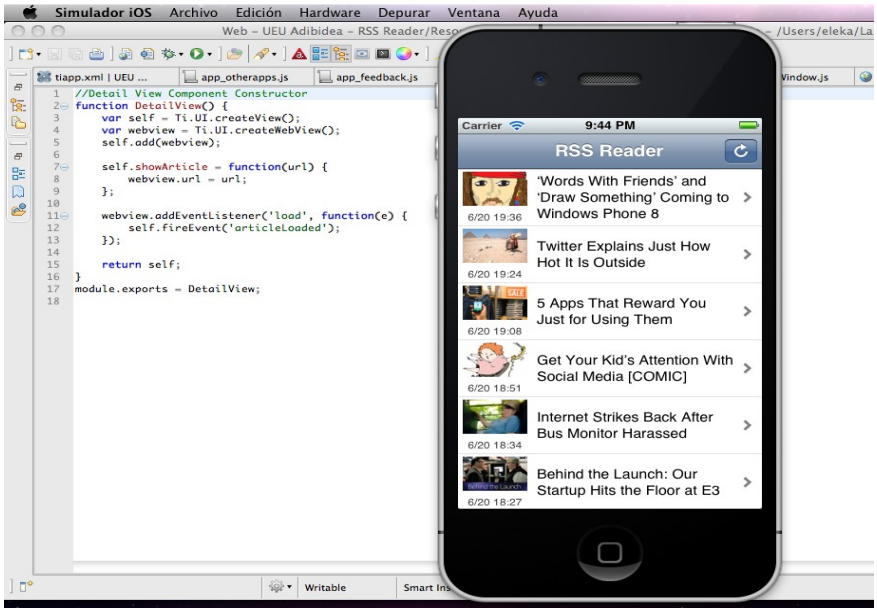
3. Proiektuari izena jarri, esaterako, *RSS Reader Adibidea*:



4. *Finish* sakatu eta berehala proiektua inportatuko da



5. Aplikazioa emuladorean edo edozein mugikorretan martxan jartzeko moduan dago.



Hauexek dira Titanium plataformarekin batera datozen adibide batzuk (http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Example_Applications):

Kitchen Sink

Adibide honek Titanium Mobile APIaren funtzionalitate gehienak erabiltzen ditu. Kontutan izan Kitchen Sink aplikazioak ez dituela jarraitzen programatzeko praktika onak, beraz, bere egitura ez litzateke erabili beharko aplikazio erreal batean.

Ondoren datozen aplikazioen egiturak erabili adibide bezala, aplikazio erreal bat sortzerako garaian.

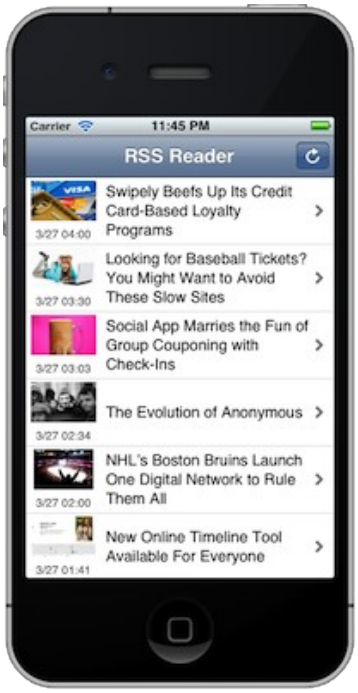
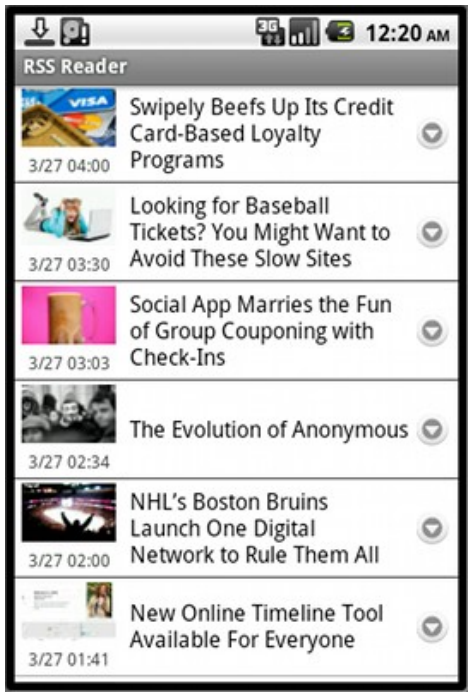
RSS Reader

Adibide honekin RSS jarioak nola kontsultatu azaldu nahi da. Lehendabizi Internetetik RSS jarioa irakurri eta tituluak eta thumbnail irudiak erakusten ditu. Behin bat hautatuta, artikulu osoa irakur daiteke.

Adibide honen bitartez erakutsi nahi dena:

- Urruneko datuen atzipena `Ti.Network.HTTPClient` erabiliz
- Javascript modularra CommonJS erabiliz
- Listak osatzen `Ti.UI.TableView` erabiliz

- Web edukiak erakusten `Ti.UI.WebView` erabiliz
- Android menua
- iOS nabigazio kontrola `Ti.UI.iPhone.NavigationGroup` erabiliz
- Plataforma-gurutzatu diseinua[3]

iOS	Android
	

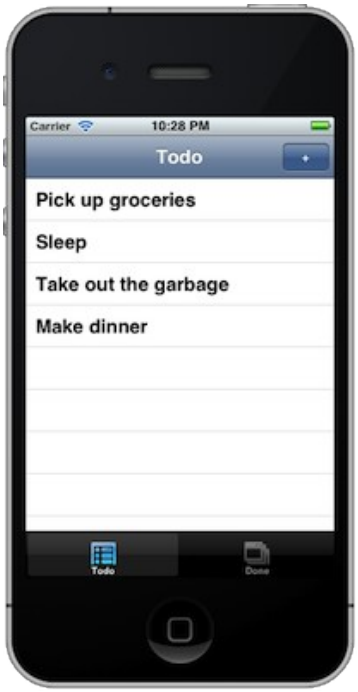
Todo List

Todo List fitxekin osatutako aplikazio bat da zereginen zerrenda baten kudeaketa sinple bat egiten duena.

Adibide honen bitartez erakutsi nahi dena:

- Biltegiratze lokala SQLite `Ti.Database` erabiliz.
- Javascript modularra CommonJS erabiliz
- Android menua
- iOS navigation bar buttons
- Plataforma-gurutzatu diseinua

3 Cross-plataform design

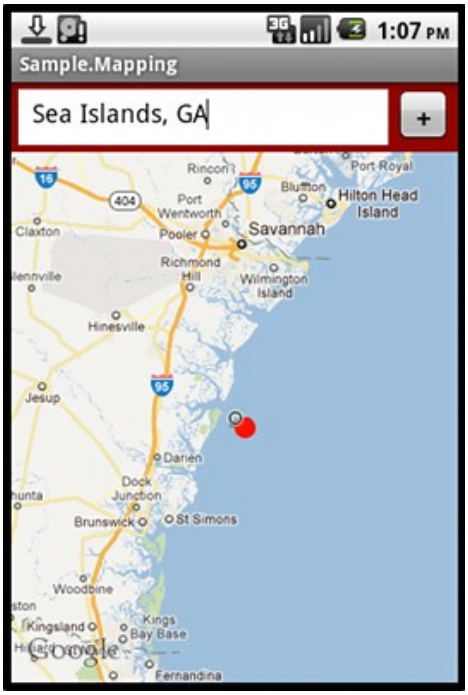
iOS	Android
	

Geocoder

Titanium Mobile adibide honek mapa natiboak erabiltzen ditu kokapen batzuk markatzeko; helbideak mapan gehitzeko aukera ematen du, anotazio moduan.

Adibide honen bitartez erakutsi nahi dena:

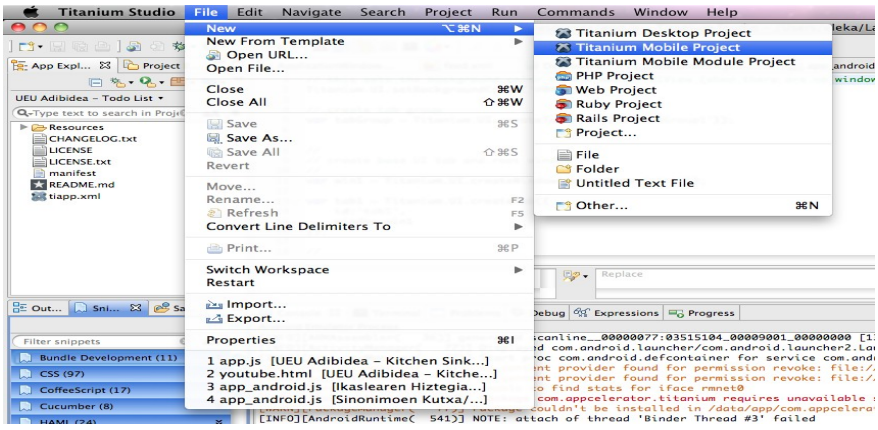
- Mapa natiboak integratzen `Titanium.Map` erabiliz
- Mapa natiboei anotazioak gehitzen
- Urruneko datu atzipena `Titanium.Network.HTTPClient` erabiliz
- Javascript modularra CommonJS erabiliz
- Plataforma-gurutzatu diseinua

iOS	Android
	

2.1.6 Lehen aplikazioa sortzen

Aplikazioaren egitura nolakoa den kontutan izanda, aplikazio bat sortzerako garaian, proiektua hutsetik hasi behar dugun edo Titanium Studiok eskaintzen digun txantiloi bat eredu bezala erabili nahi dugun erabaki beharko dugu.

1) Proiektu berria sortzeko, aukeratu *File-> New -> Titanium Mobile Project*



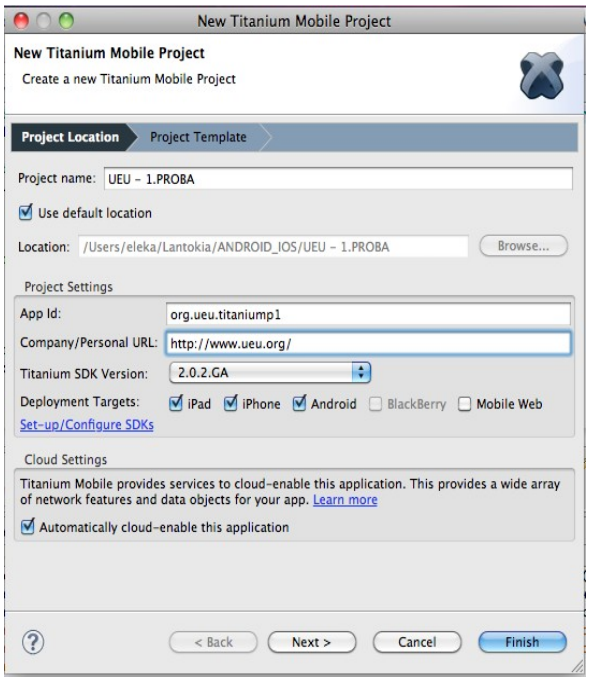
2) Proiektuaren informazioa sartu. Hauek dira beharrezko datuak:

Project Name: Proiektuaren izena

App Id: Aplikazioaren identifikadorea (adibidez, org.eu.ariketa1). Bere formatua internet helbidearen kontrakoa (*reverse domain name format*, adibidez, org.ueu) eta bukaeran aplikazioaren izena (esaterako, *ariketa1*)

Company/Personal URL: Pertsonaren edo enpresaren URLa (adibidez, http://www.ueu.org)

Titanium SDK Version: Hemen Titanium-en SDKren bertsioa jarriko dugu. Momentu honetan, azkena, 2.0.2.GA.



3) Next botoia sakatu

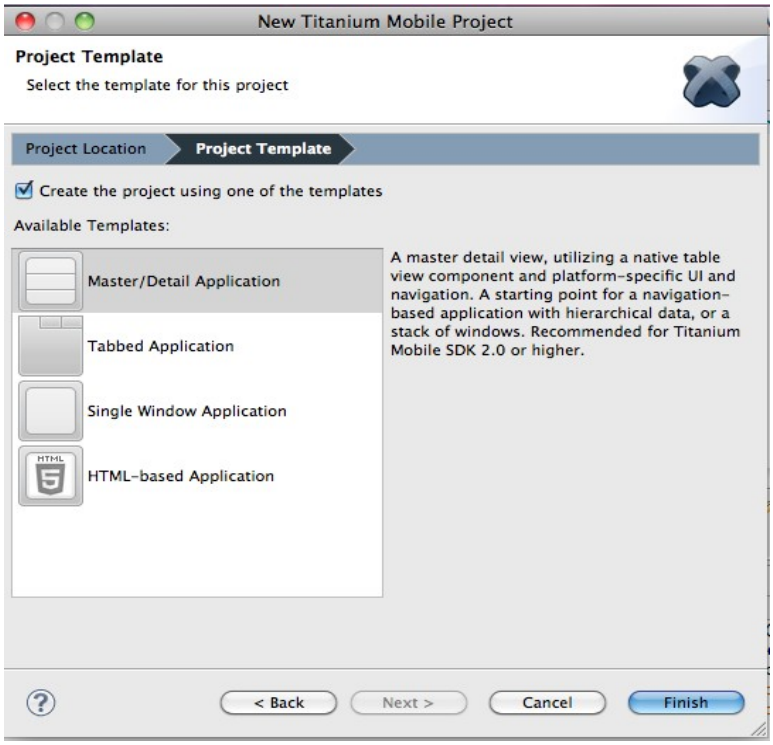
4) Sortu behar duzun aplikazioaren arabera, txantiloi bat hautatu.

Master/Detail application: Zerrenda bat eta osagai batean klik egin ondoren bere detailea erakusten duen aplikazioa

Tabbed application: Fitxak dituen aplikazioa

Single Window application: Leiho bakarra duen aplikazioa

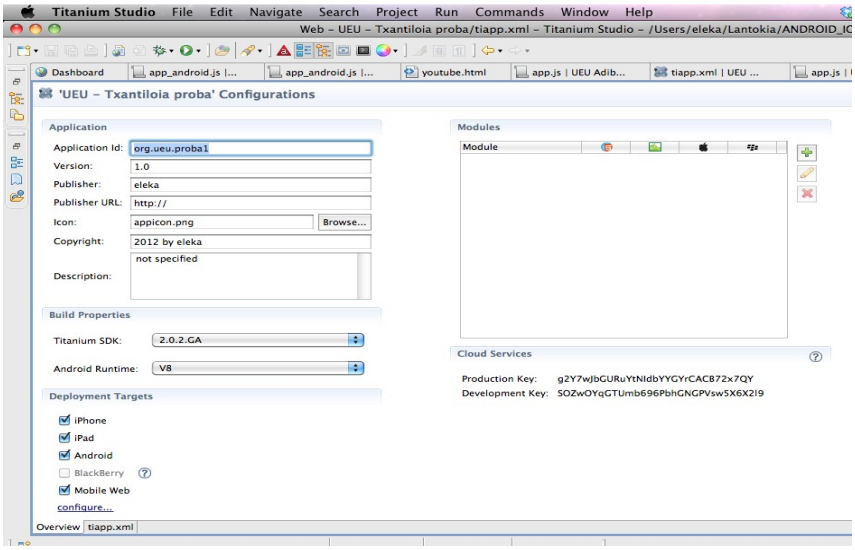
HTML-based application: HTML bista bat erabiltzen duen aplikazioa



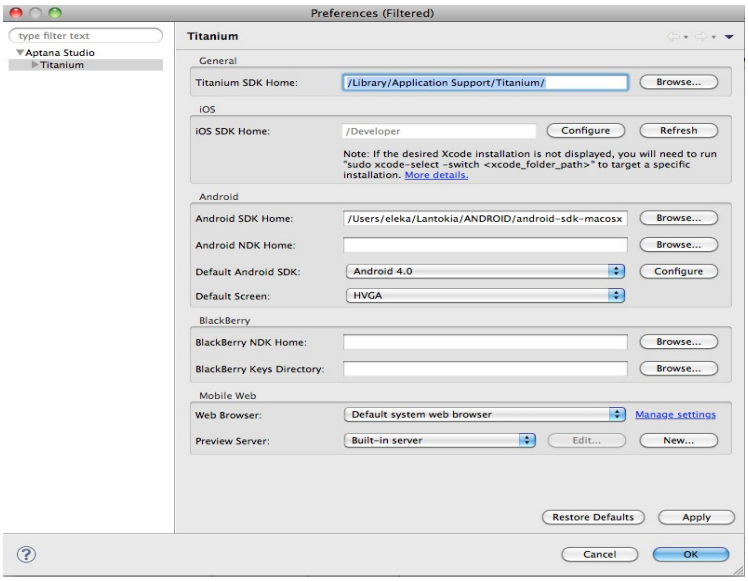
2.1.7 Lehen aplikazioa moldatzen

Behin aplikazioa sortu dugula, konfigurazioa, ikonoak eta kodea moldatzen has gaitezke.

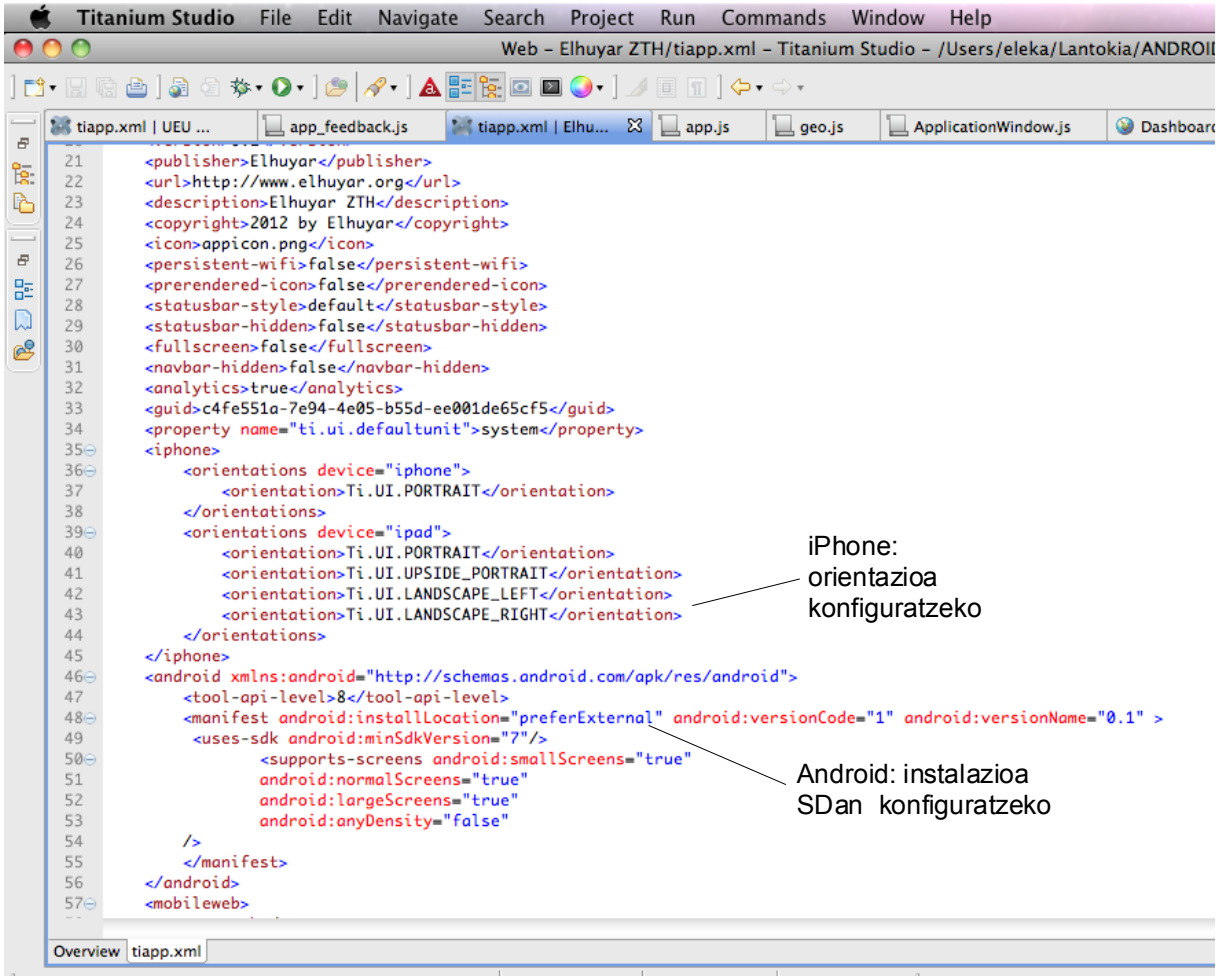
1) Aplikazioa konfiguratzeko, *tiapp.xml* fitxategia editatu behar da. Beraren gainean klik egitean, defektuz formulario itxurako leihoa agertuko zaigu datuak aldatzeko.



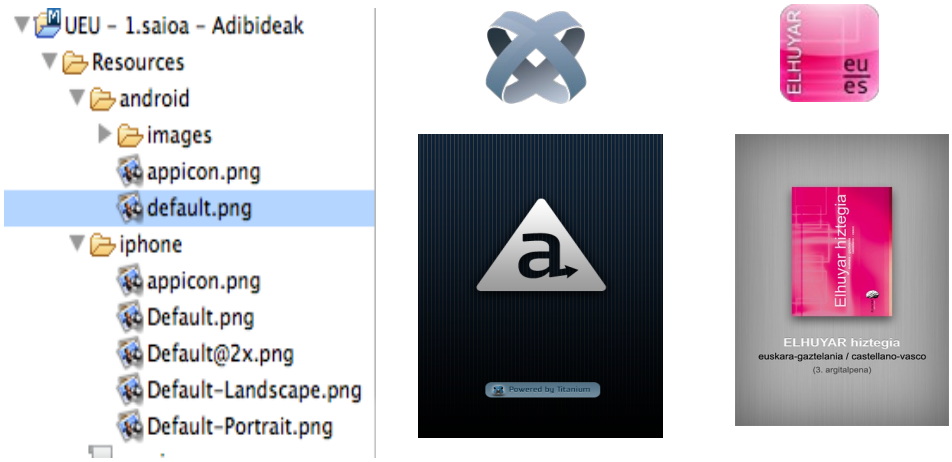
2) Bertan, aurrez aipatutako informazioaz gain, aplikazioaren bertsioak, deskribapena etab. osatu. SDK ezberdinak ere konfiguratu daitezke, leihoaren azpikaldean dagoen *configure...* estekan klik eginez gero.



3) Bere .xml bertsioa zuzenean editatu daiteke, modu horretan, aplikazioa gehiago konfiguratu daitekeelarik. Adibidez, iPhone/iPad-en kasurako zein orientazio onartuko dituzten konfigura daiteke (posible da iPhonearen kasuan etzana ez onartzea) eta Android gailuen kasuan, aplikazioa defektuz SDan instalatzeko agindua jar daiteke bertan.



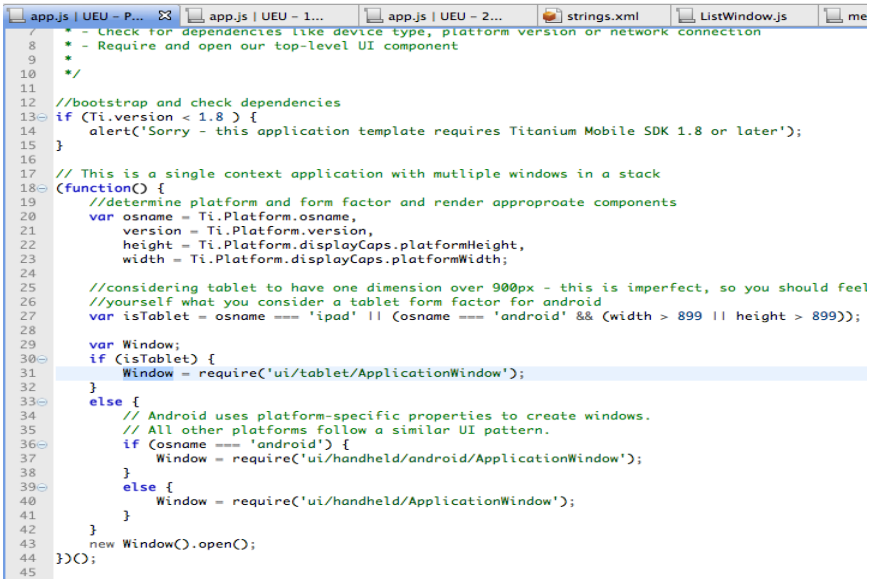
4) Txantiloiak defektuz ekartzen dituen ikono (gailuaren mahaigainean agertzen direnak) eta splash pantailak (aplikazioa martxan jartzen denean agertzen den pantaila) ordezkatu, tamainak errespetatuz.



Informazio gehiago:

http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Icons_and_Splash_Screens

5) Lehendabizi exekutatzen den Javascript kodea app.js fitxategian dagoena da. Bertan egiten da lehen konprobaketa, zein sistema eragile (Android/iOS) eta gailu motaren (eskuko telefonoa/tableta) gainean lanean ari garen jakiteko. Hemendik aurrera, programatzeko jardunbide egokiak erabiliz (ikusi 2.2.2 atala), aplikazioko kodea moldatuz joango gara, gure aplikazioak behar dituen funtzionalitateak gehituz.



2.1.8 Praktika

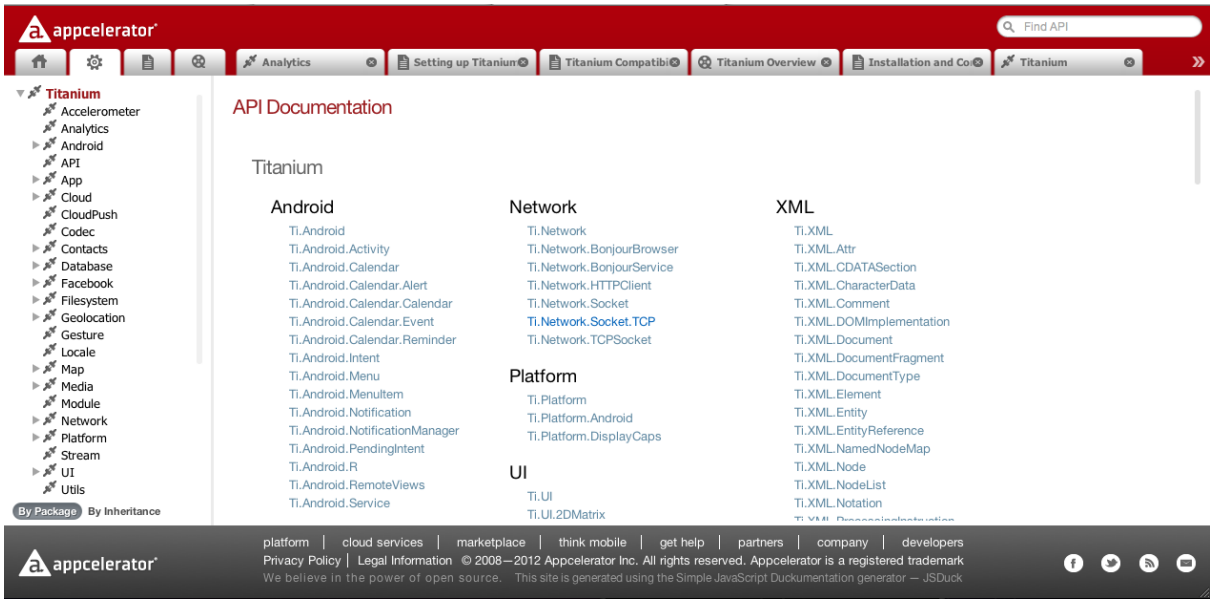
1) Appcelerator Network kontua sortu
2) Titanium plataformaren instalazioa: instalatu edo errepasatu
3) Adibideak inportatu (*Kitchen Sink*, *Geocoder*, *Todo List* eta *RSS reader*) eta martxan jarri (*Geocoder, Todo List, RSS Reader*)
4) Txantiloi ezberdinak erabiliz, proiektuak sortu eta emuladorean martxan jarri, moldaketarik egin gabe (Single Window, Tabbed, Master/Detail)
5) Moldatu aurreko ariketan *Single Window* txantiloiarekin sortutako proiektuaren *tiapp.xml* fitxategia aplikazioaren bertsioa 1.2 izan dadin eta SD txartelean gordea izan dadin.

2.2. Aplikazio bat sortzen (I)

2.2.1 Titanium APIa

Titanium Mobilen API osoa hementxe kontsulta daiteke[4]:

http://docs.appcelerator.com/titanium/2.0/index.html#!/api



Titanium modulu nagusia, 25 azpimodulutan banatzen da, ondorengoak direlarik:

Ti.Accelerometer	Ti.Analytics	Ti.Android	Ti.API	Ti.App
Ti.Cloud	Ti.CloudPush	Ti.Codec	Ti.Contacts	Ti.Database
Ti.Facebook	Ti.Filesystem	Ti.GeoLocation	Ti.Gesture	Ti.Locale
Ti.Map	Ti.Media	Ti.Module	Ti.Network	Ti.Platform
Ti.Stream	Ti.UI	Ti.Utils	Ti.XML	Ti.Yahoo

Horietako modulu batzuk, nabarmenduta daudenak hain zuzen, ikastaroan zehar landuko ditugu.

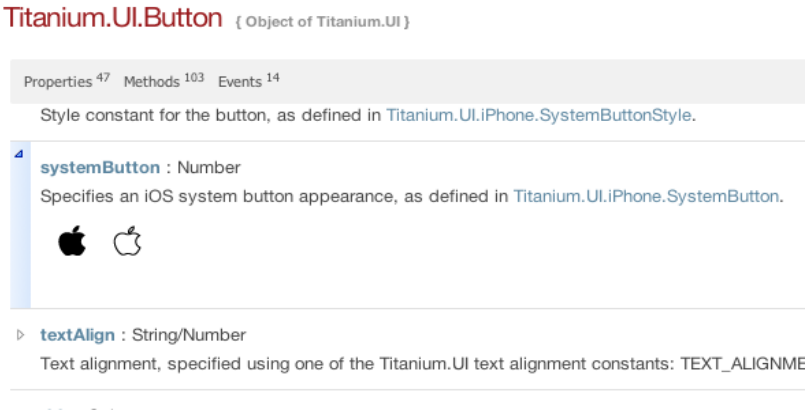
Modulu bakoitzaren dokumentazioa aztertuz gero, bakoitzaren propietateak, metodoak eta gertaerak ikus ahal izango ditugu, baita adibideak ere, hainbat kasutan. Har dezagun adibide modura `Ti.UI.Button` objektua azalpenak emateko.

4 Ikastaroa ematerako garaian 2.0 bertsioa dago eskuragarri eta bertsio horretaz arituko gara

Propietateak

Objektuari ezaugarriak esleitzeko erabiltzen dira. Button objektuak, esaterako 47 propietate ditu. Horien artean: *top* (altuera), *backgroundColor* (atzeko-planoko kolorea), *width* (zabalera), *font* (letra-mota)...

Propietate bat plataforma guztietarako existitzen ez bada, funtzionatzen duen plataformetako ikonoa agertzen da propietatearen deskribapenaren ostean. Adibidez:



Metodoak

Botoia sortzeko `Titanium.UI.createButton` metodoa erabiliko dugu. Modu honetan:

```
var button = Titanium.UI.createButton({ title: 'Hello' });
```

Titanium.UI.Button objektuak, 103 metodo ditu bere baitan. Horien artean: *getSize* (botoiaren tamaina lortzeko), setHeight (zabalera ezartzeko)...

```
var tamaina = button.getSize();
```

Metodoen artean berezia eta askotan erabiltzen dena, ***addEventListener****(String name, Callback<Object> callback)* metodoa da.

```
button.addEventListener('click',function(e) {
      Titanium.API.info("Botoia sakatu duzu!");
});
```

Metodo honen bitartez, button botoiaren *click* gertaera suertatzen denean, *callback* parametroak dagoen funtzioa exekutatuko da. Kasu honetan, “Botoia sakatu duzu!” mezua agertuko da kontsolan.

Gertaerak

Button objektuak 14 gertaera ditu, zehazki. Horien artean, goian aipatutako *click* (klik egitean), *longpress* (luzaro botoia sakatzean)... Gertaera hauek tratatzeko, goian aipatutako metodoa erabiltzen da (*addEventListener*).

2.2.2 Programatzeko jardunbide egokiak

Aurreko atalean aipatu dugu nola adibide batzuk Javascript modularra erabiltzen duten. Titanium Mobile-n inplementazioa CommonJS module espezifikazioetan oinarrituta dago, beren aplikazioek egitura egokia izan dezaten. Zehazki, node.js inplementazioan oinarritua.

Definizio batzuk, atal hau hobe ulertzeko:

- **Module**: CommonJS moduluari deituko diogu. Aplikazio beraren Javascript fitxategi bat izan daiteke, edo Titanium-en hedapen natibo bat, Javascript API bat eskaintzen duelarik.
- **Resources**: Prozesatu edo konpilatu aurretik, garatzailearen iturburu-kodea gordetzen den karpeta.
- `exports` : Modulu barruko aldagai askea. Interfaze publiko bat sortu ahal izateko, propietate ezberdinak esleitu dakizkioke.
- `module.exports`: Modulu barneko objektua, zeina moduluaren interfaze publikoa eskainiko duen objektuarekin ordezkatua izango den.

Erabilera

Titanium-eko modulu bat erabiltzeko, `require` funtzioa erabili behar da

Esaterako:

```
var nireModulua = require('kaixoEsanModulua');
```

`require` funtzioari pasatako argumentuak bat etorri behar du Titanium-ek atzitu dezaken modulu baten izenarekin. Funtzio honen emaitza bezala Javascript objektu bat lortuko dugu, propietate eta funtzioekin. Adibidez, *kaixoEsanModulua*-k *kaixoEsan* izeneko funtzio bat esportatzen du, zeinak kontsolan mezu bat idatziko duen “kaixo” esanez.

```
var nireModulua = require('kaixoEsanModulua');
nireModulua.kaixoEsan('Aritz');
// kontsolan idatziko du “Kaixo Aritz!”
```

Modulu natiboak/Javascript moduluak

`require` funtzioa azaltzen denean, Titanium-ek erabaki behar du modulu hori natiboa den

edo *Resources* karpetan dagoen.

Modulu natiboak kate sinple batekin identifikatzen dira, *tiapp.xml* fitxategian. Adibide bat:

```
<modules>
<module version="1.0">ti.paypal</module>
</modules>
```

Honela deituko genioke Titanium Mobile aplikazio batetatik:

```
var paypal = require('ti.paypal');
```

Beraz, Titanium-ek badaki *ti.paypal* modulu natibo bat dela eta ez da hasiko Resources karpetan ezeren bila, baldin eta modulu hori topatzen ez badu.

Javascript moduluak *Resources* karpetatik hartuko ditu. Fitxategiaren bidea zehaztu behar da, baina *.js* luzapenik gabe:

```
var myModule = require('../lib/myModule');
```

exports

Modulu barruko aldagai askea. Interfaze publiko bat sortu ahal izateko, propietate ezberdinak esleitu dakizkioke. Adibidez:

kaixoEsanModulua.js

```
exports.kaixoEsan = function(name) {
Ti.API.info('Kaixo '+name+'!');
};

exports.version = 1.4;
exports.author = 'Iker Agirre';
```

app.js

```
var nireModulua = require('kaixoEsanModulua');
nireModulua.kaixoEsan('Aritz');
```

// kontsolan idatziko du “Kaixo Aritz!”

module.exports

Modulu barneko objektua, zeina moduluaren interfaze publikoa eskainiko duen objektuarekin ordezkatua izango den. Adibidez:

Person.js

```
function Person(firstName,lastName) {
    this.firstName = firstName;
```

```
    this.lastName = lastName;
}
Person.prototype.fullName = function() {
    return this.firstName+' '+this.lastName;
};
module.exports = Person;
```

app.js

```
var Person = require('Person');
var don = new Person('Don','Thorp');
var donsName = don.fullName(); // "Don Thorp"
```

Programatzeko jardunbide egokiak jarraitzen ikasteko, aztertu *Titanium Studio*rekin batera datozen txantiloiak, nahiz adibideak (*KitchenSink* izan ezik)

Informazio gehiago:

https://wiki.appcelerator.org/display/guides/Best+Practices

http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Best_Practices_and_Recommendations

2.2.3 Oinarrizko interfazeak

Atal honetan nabigazioa, leihoak, bistak eta fitxei buruzko oinarrizko azalpenak emango dira. Interfazearekin lotuta dagoen oro, `Titanium.UI` moduluan topatuko dugu. Gainera, badira azpimodulu espezifiko batzuk, soilik smartphone sistema eragile bakoitzarentzat bereziki sortuak: `Titanium.UI.Android`, `Titanium.UI.iPhone`, `Titanium.UI.iPad` izenekoak.

Leihoa

Titanium aplikazio baten oinarrizko elementua leihoa da. Leihoa edukiontzi bat da, nahi adina elementu bertan jartzeko (botoiak, irudiak, etab.). Aplikazio batean leiho bat baino gehiago egon daiteke, baina gutxienez bat. Bere kabuz sor daitekeen elementua da, inoren umea izan gabe.

Adibidea:

1.aukera

```
var win1 = Titanium.UI.createWindow({
    title:'Tab 1',
    backgroundColor:'#fff'
});
```

2.aukera

```
var win1 = Titanium.UI.createWindow();
win1.title='Tab 1'
win1.backgroundColor='#fff';
```

Informazio gehiago: `Ti.UI.Window`

Bista

Titaniumekin sortutako aplikazio baten beste oinarrizko elementua da eta web programatzaile baten ikuspegitik, HTMLz etiketa baten analogoa izan daiteke. Hitz gutxitan, laukizuzen moduko bat da, bertan elementu ezberdinak mantentzeko eta aplikazioaren kontrola errazteko. Bista bat ezin da bere kabuz soilik existitu. Derrigorrez, leiho baten edo bista elementua ume bezala onartzen dituen elementu baten umea izan behar du (TableRowView bat edo beste bista bat, adibidez)

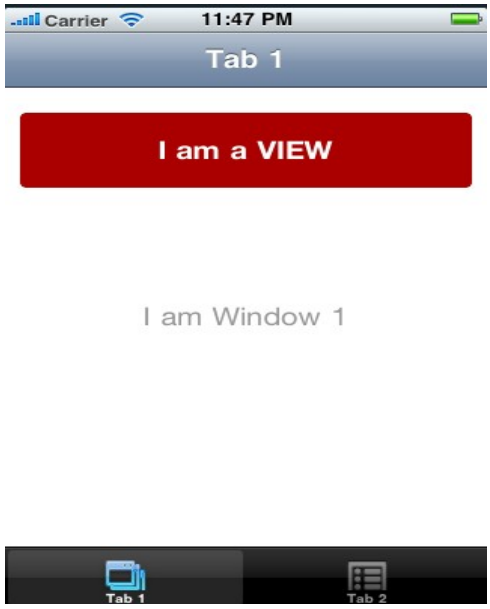
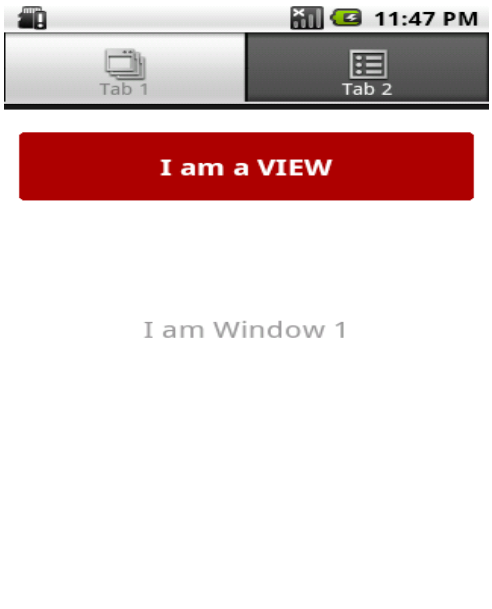
Informazio gehiago: `Ti.UI.View`

Fitxak

Oinarrizko nabigazio sistema osatzeko TabGroup eta Tab elementuak erabiltzen dira. Fitxak dituen web aplikazio batean bezalako funtzionalitatea eskaintzen diote aplikazioari. Fitxa bakoitzak, leiho bat kontrolatzen du. Aplikazioak *TabGroup* bat erabili behar duela erabakitzen badugu, elementu hauxe izango da aplikazioaren erroa. Zer esan nahi dugu honekin? Tabgroup baten bidez irekitako edozein leiho honengatik kontrolatua izango dela eta TabGroup-a ezin dela itxia edo ezabatua izan, soilik ezkutatua.

Gainera, atal honetan aipa dezakegu, badela iOS sistemaren kasurako nabigazio sistema espezifiko bat, NavigationGroup izenekoa. Honen bitartez, erabiltzaileari leiho ezberdinetan zehar nabigatzeko aukera eskaintzen diogu garatzaileak horretarako espreski barra eta botoiak sortu beharrik gabe.

Informazio gehiago: `Ti.UI.TabGroup, Ti.UI.Tab, NavigationGroup`

Iphone	Android
	

Iturria: http://cssgallery.info/seven-days-with-titanium-day-1-windows-views-navigation-and-tabs/

Taulak

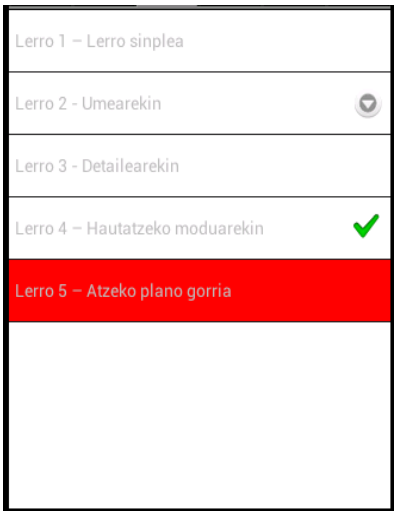
Aplikazioak garatzerako garaian elementu oso erabilgarria da, gainera iPhone-n kasuan gehien erabiltzen den IU elementua dela diote. Kasu batzuetan oso nabarmena da erabiltzen ari garela, baina beste batzuetan modu ezkutuan erabiltzen dugu.

Elementu honetako propietate garrantzitsuena *data* propietatea da, izan ere, hau bete gabe elementu honek ez baitu zentzu handirik.

Ikusi ondorengo adibide hau:

```
var win1 = Titanium.UI.createWindow({
backgroundColor:"#fff"
});
var table1 =  Titanium.UI.createTableView({
data:[
        {title:"Lerro 1 - Lerro sinplea"},
        {title:"Lerro 2 - Umearekin", hasChild:true},
        {title:"Lerro 3 - Detailearekin", hasDetail:true},
        {title:"Lerro 4 - Hautatzeko moduarekin", hasCheck:true},
        {title:"Lerro 5 - Atzeko plano gorria", backgroundColor:"#f00"}
]
});
win1.add(table1);
win1.open();
```

Ondorengo hau erakutsiko luke (Android/iPhone):

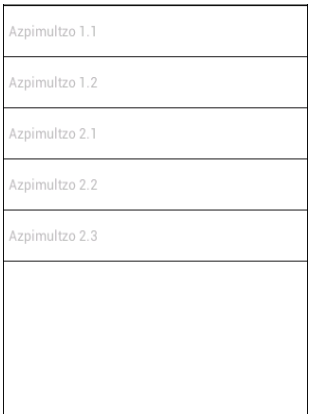
iPhone	Android
	

Hauxe da taulak sortzeko modurik sinpleena, lerro bakoitzeko, Titanium.UI.TableViewRows elementuaren propietateak kontutan hartuz.

Taulak multzokatu daitezke, `Titanium.UI.createTableViewSection()` metodoa erabiliz (iOS sistema). Adibidea:

```
var  table1 =  Titanium.UI.createTableView({
style:Titanium.UI.iPhone.TableViewStyle.GROUPED
});
var section1 = Titanium.UI.createTableViewSection();
section1.headerTitle = "Multzo 1";
var row1 = Titanium.UI.createTableViewRow({title:"Azpimultzo 1.1"});
var row2 = Titanium.UI.createTableViewRow({title:"Azpimultzo 1.2"});
section1.add(row1);
section1.add(row2);
var section2 = Titanium.UI.createTableViewSection();
section2.headerTitle = "Multzo 2";
var row3 = Titanium.UI.createTableViewRow({title:"Azpimultzo 2.1"});
var row4 = Titanium.UI.createTableViewRow({title:"Azpimultzo 2.2"});
var row5 = Titanium.UI.createTableViewRow({title:"Azpimultzo 2."});
section2.add(row3);
section2.add(row4);
section2.add(row5);
table1.setData([section1,section2]);
```

Hauxe ikusiko litzateke (Android-en ez da multzokatuta ikusten):

iPhone	Android
	

Informazio gehiago:

`Ti.UI.TableView, Titanium.UI.TableViewRow, Titanium.UI.TableViewSection`

Taula pertsonalizatuak egiteko: http://nosoloweb.es/filas-personalizadas-para-tableviews-con-titanium

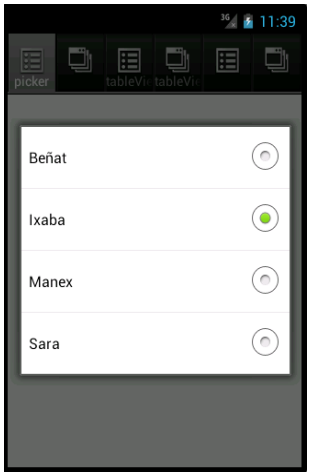
Hautagailuak

Hautagailua (picker) HTMLz <*select*> etiketaren analogoa litzateke, desberdintasun bakarra zera da, Titanium erabiliz kolore-anitzeko hautagailua egin daitekeela (dirudienez, hau ez da posible HMTLz egitea, non eta ez diren truko zehatzak erabiltzen). Hautagailu bat sortzeko `Titanium.IU.createPicker` metodoa erabili behar dugu.

Adibide bat:

```
var picker = Titanium.UI.createPicker();
var data = [];
data.push(Titanium.UI.createPickerRow({title:'Beñat'}));
data.push(Titanium.UI.createPickerRow({title:'Ixaba'}));
data.push(Titanium.UI.createPickerRow({title:'Manex'}));
data.push(Titanium.UI.createPickerRow({title:'Sara'}));
picker.add(data);
```

Hauxe ikusiko genuke:

IPhone	Android
	

Modulu honen type propietatea erabiliz aurrez sortutako hautagailuak erraz sor ditzakegu:

```
Titanium.UI.PICKER_TYPE_PLAIN (lehenetsia),
Titanium.UI.PICKER_TYPE_DATE_AND_TIME
Titanium.UI.PICKER_TYPE_DATE, Titanium.UI.PICKER_TYPE_TIME
Titanium.UI.PICKER_TYPE_COUNT_DOWN_TIMER
```

Interfazeak osatzeko beste elementu batzuk:

Nahiz eta ez dituen praktika onak erakusten, KitchenSink adibidearen Base UI eta Controls fitxetan elementu ezberdinak ikusteko aukera eskaintzen da:

```
Titanium.UI.Button
Titanium.UI.Slider
Titanium.UI.SearchBar
Titanium.UI.Label
Titanium.UI.TextField
...
```

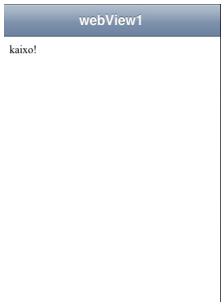
2.2.4 Web edukiak

`Webview` elementua web nabigatzaile bat soilik da, baina ez ditu ohiko nabigatzaileek dituzten kontrolak. `Titanium.UI.WebView` elementuaren bitartez web orri lokalak nahiz urrunekoak erakutsi ahal izango dira apikazioan.

WebView elementua HTML/CSS erabiliz osatutako orri paregabeak erakusteko erabili daiteke, baina kontutan izan behar dugu hori ez dela oso eraginkorra gure gailularentzat (CPU eta memoria erabilpenagatik), beraz, beharrezkoa den kasuetan soilik erabiltzea gomendatzen da.

`WebView`-a `Titanium.UI.createWebView` metodoaren bidez sortzen da (beste hainbat elementu bezala) eta zerbait aipatzekotan, esan daiteke bere gertaerak kudeatu daitezkeela. Esaterako, interesgarria izan daiteke *beforeLoad* eta *Load* gertaerak tratatzea, web orria kargatzen den bitartean erabiltzaileari "Itxaron" leihoa erakusteko.

Zuzenean HTMLa erakusteko, ondorengo adibidea:




```
var webview =
Titanium.UI.createWebView( {html:'<html><body>kaixo!
</body></html>'});
var window = Titanium.UI.createWindow();
window.title = "webView1";
window.add(webview);
window.open();
```

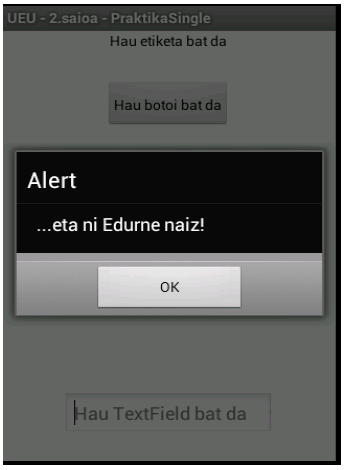
Urruneko URLak erakusteko:


```
var webview = Titanium.UI.createWebView(
{url:'http://www.ueu.org'});
var window = Titanium.UI.createWindow();
window.title = "webView2";
window.add(webview);
window.open();
```

Informazio gehiago: `Ti.UI.WebView`

2.2.5 Praktika

1) Sortu *Single Window* txantiloietik abiatuz ondorengo irudian agertzen den bezalako leiho bat. Edozein elementutan klik eginez gero, mezu bat agertarazi:



2) Sortu aplikazio bat web gune zerrenda bat erakutsiko duena eta zerrendako osagai bat hautatu ondoren bere informazio guztia erakutsiko duena (izena, herria, url-a, web edukia). Fitxari ahalik eta formatu egokiena ematen saiatu, goiburu bezela izena erakutsiz eta goiburuaren fondo bezala taulan ezarritako kolorea ezarriz. Hauxe da erakutsi beharreko informazioa:

Taularen titulua: INFORMAZIOA

Izena	Herria	URL-a	Kolorea
UEU	Eibar	http://www.ueu.org	Urdina
Eleka	Usurbil	http://www.eleka.net	Berdea
Elhuyar	Usurbil	http://www.elhuyar.org	Gorria
Eibarko Udala	Eibar	http://www.eibar.es	Horia

3) Multzokatu zerrendak herriaren arabera. Bi modutara egin daiteke: nabigazioa bikoiztuz edo `TableViewSection` erabiliz.

2.3. Aplikazio bat sortzen (II)

2.3.1 Urruneko datuak

Zerbitzariarekin maila baxuago batean komunikatu nahi badugu (API mailan, zure zerbitzariaren garapen propio batekin...), `Ti.Network.HttpClient` erabili dezakezu, zeina *XHR*en (edo *Ajax*) parekoa den web teknologian. `HttpClient` erabiltzen dugunean hauxe eduki behar dugu kontutan:

– Deia modu asinkronoan egiten dela, beraz, modu horretarako prestatu behar duzula zure aplikazioa

– Sekuentzia bat jarraitu behar duzula: sortu, gertaerak ezarri, ireki eta bidali

Beraz, `HttpClient` erabiltzean *onload* eta *onerror* gertaerak ezarri behar dituzu, kanala ireki eta eskaera egin behar duzu (parametro edo parametrorik gabe).

```
var xhr = Titanium.Network.createHTTPClient();
xhr.onload = function()
{
// this.responseText holds the raw text return of the message (used for JSON)
// this.responseXML holds any returned XML (used for SOAP web services)
// this.responseData holds any returned binary data
Ti.API.info(this.responseText);
Ti.API.info('loaded');
};
xhr.onerror = function()
{
Titanium.API.info('error');
};
xhr.open("GET","http://google.com");
xhr.send();
```

onload gertaerarekin erabilitako funtzioan *responseText* (JSON erantzuna), *responseXML* (XML erantzuna) edo *responseData* (erantzun binarioa) erabil ditzakezu erantzuna prozesatzeko.

Twitter APIa erabiliz Twitter bezero bat nola sortu erakusten duen adibidea: Appcelerator: Using JSON to Build a Twitter Client (kontutan izan ez dituela jardunbide egokiak erabiltzen, CommonJS modulartasunari dagokionez).

2.3.2 Media: Irudiak, soinuak eta pelikulak

Irudiak

Irudi bat erakusteko `Ti.UI.ImageView` erabiltzen dugu. Elementu honek betiko argumentuak erabiltzen ditu (*top, left, width...*), baina bada bat elementu honek bereziki duena, *image* izenekoa. Bertan, erakutsi nahi dugun irudiaren jatorria jarri behar dugu, fitxategiak lokalak nahiz urrunekoak izan daitezkeelarik. Adibidez:

```
var the_img = Titanium.UI.createImage({
image:"test.jpg"
})
```

Irudi lokalak erakutsi ahal izateko, *Resources* karpetaren barruan egon behar dute eta image argumentuan jartzen den bidea erlatiboa izango da. Urruneko irudien kasuan, berriz, URLa jartzearekin nahikoa da. Adibidez:

```
var ueu_logoa = Titanium.UI.createImage({
image:"http://www.ueu.org/img/logo.png"
})
```

Soinuak

Gailuan bertan ditugun soinuak erabil ditzakegu gure aplikazio nahiz jokuetan txertatzeko. Soinuak modu honetan kargatzen dira:

```
var soinua = Titanium.Media.createSound({
url:'sound.wav'
});
```

Soinuak erreproduzitu, gelditu, berrasieratu eta bolumena kudeatu dezakegu `Titanium.Media.Sound` elementuaren metodoak erabiliz. Soinu bakoitzak bere gertaerak ditu, horietako bat complete da, zeina soinuak erreproduzitzeari uzten dionean jazotzen den.

Soinua oso pisutsua bada, komeni da sortzerako garaian preload propietatea true jartzea, bestela gerta baitaiteke play-ri eman orduko soinua martxan ez jartzea.

Azkenik, aipatzeko beste kontu bat, komeni dela ere soinua askatzea gehiago erabili behar ez bada. Adibidez, soinu bat aplikazioa martxan jartzen denean soilik erabiltzen badugu, erreproduzitzen amaitu bezain laster, komeni askatzea, gailuaren memorian ere lekua egiteko.

Horrela egin daiteke:

```
var sarrera_soinua = Titanium.Media.createSound({
url:'cricket.wav',
preload:true
});
intro_sound.addEventListener('complete', function(e)
{ intro_sound.release(); });
intro_sound.play();
//... kodea
```

Goiko kodeak soinua erreproduzitzen du eta automatikoki askatzen du erreproduzitzen amaitu orduko.

Bideoak

Bideoak erabiltzea soinuak erabiltzearen modukoa da, baina `Titanium.Media.VideoPlayer` erabiliz. Kasu honetan erreproduktore bat sortu beharko dugu `Ti.Media.createVideoPlayer` metodoa erabiliz.

Adibidez:

```
var videoURL = 'http://vimeo.com/26414926';
var activeMovie = Titanium.Media.createVideoPlayer({
url: videoURL,
width:300,
height:200,
top:50,
left:50,
backgroundColor:'#0f0'
});
view.add(activeMovie);
activeMovie.play();
```

Kasu honetan ere bideoa fitxategi lokal bat edo urrunekoa izan daiteke, baina kontutan izan behar dugu sistema eragileak zeintzuk onartzen dituen (begiratu horretarako VideoPlayer elementuaren dokumentazioa).

iOS 3.2 ondorengo SDKn alde ona zera da: ez dela beharrezkoa bideoa pantaila osoan erreproduzitzea, hau da, guk nahi dugun tamainako bista batean erakutsi dezakegu bideoa.

2.3.3 Kokapena: Geolokalizazioa eta mapak

Geolokalizazioa

Aplikazio bat martxan jartzen duzunean, koordenatuak gailuaren cachean aurkitzen direnak dira, beste hitz batzuetan esanda, azken aldiz geolokalizazioa erabili zenean detektatu zirenak. Momentuko kokapena zein den jakiteak bere kostua du, beraz, gure aplikazioa geolokalizazioan oinarriturik badago, ezinbestez hartu behar dugu hau kontutan eta neurriak hartu. `Titanium.Geolocation` elementuak eskaintzen ditu geolokalizazioa maneiatzeko beharrezko metodo guztiak.

Gehien erabiltzen den metodoa *getCurrentPosition* da, zeinak momentuko posizioa zein den adieraziko duen eta *location* gertaera jazoko duen. Azken gertaera hau kokapenez aldatzen garen bakoitzean jazotzen da. Beraz, demagun erabiltzaile batek egiten duen bidea erregistratu nahi dugula. Horretarako, aplikazioa martxan jartzen denean, bere hasierako posizioa lortu eta ibiltzen doan heinean, denbora tarte jakin bakoitzeko, posizio puntuak lor ditzakegu eta horren arabera bidea

markatu.

```
	Titanium.Geolocation.distanceFilter = 10; // set the granularity of the
location event
	Titanium.Geolocation.getCurrentPosition(function(e)
	{
		if (e.error)
		{
			// errorea kudeatu
			return;
		}
		var longitude = e.coords.longitude;
		var latitude = e.coords.latitude;
		var altitude = e.coords.altitude;
		var heading = e.coords.heading;
		var accuracy = e.coords.accuracy;
		var speed = e.coords.speed;
		var timestamp = e.coords.timestamp;
		var altitudeAccuracy = e.coords.altitudeAccuracy;
		// datuak prozesatu behar ditugun moduan
	});
	Titanium.Geolocation.addEventListener('location',function(e)
	{
		if (e.error)
		{
			// errorea kudeatu
		return;
		}
		var longitude = e.coords.longitude;
		var latitude = e.coords.latitude;
		var altitude = e.coords.altitude;
		var heading = e.coords.heading;
		var accuracy = e.coords.accuracy;
		var speed = e.coords.speed;
		var timestamp = e.coords.timestamp;
		var altitudeAccuracy = e.coords.altitudeAccuracy;

	// berriro ere datuak prozesatu
	});
```

Informazio gehiago: KitchenSink adibidean, *GeoCoder* adibidean.

Mapak

Mapak `Titanium.Map` azpimoduluaren metodoak erabiliz sortzen dira. Mapak konplikatuxeagoak izanik ere, interfaze txukunak egiteko lagungarriak suerta daitezke. `Map.View` bat erabili dezakegu bertan bide bat markatzeko edo irudidun/tituludun/estekadun markagailuak leku konkretu baten bista zehatzago bat erakusteko.

Mapa bat sortzeko hauxe da egin beharrekoa:

```
var mapview = Titanium.Map.createView({
top:20,
height:300,
mapType: Titanium.Map.STANDARD_TYPE,
region:{latitude:33.74511, longitude:-84.38993, latitudeDelta:0.5,
longitudeDelta:0.5},
animate:true,
regionFit:true,
userLocation:true
});
```

Mapa sortzaileak argumentu desberdinak hartu ditzake, goian aipatutakoez gain, *annotations* (markagailuak), hala ere, azken hauek aurretik sortu egin beharko dira, modu honetan:

```
var apple = Titanium.Map.createAnnotation({
      latitude:37.33168900,
      longitude:-122.03073100,
      title:"Steve Jobs",
      subtitle:'Cupertino, CA',
      pincolor:Titanium.Map.ANNOTATION_GREEN,
      animate:true,
      rightButton: 'apple_logo.jpg',
      myid:2
});
var atlanta = Titanium.Map.createAnnotation({
      latitude:33.74511,
      longitude:-84.38993,
      title:"Atlanta, GA",
      subtitle:'Atlanta Braves Stadium\nfoo',
      animate:true,
      leftButton:'atlanta.jpg',
      rightButton: Titanium.UI.iPhone.SystemButton.DISCLOSURE,
      myid:3
});
```

Mapa sortzerakoan horrela pasa daitezke, aipatutako *annotations* argumentuarekin batera:

```
annotations:[atlanta,apple]
```

Bestela ere, aurrerago banaka gehitu daitezke:

```
mapview.addAnnotation(atlanta);
mapview.addAnnotation(apple);
```

2.3.4 Orientazioa eta azelerometroa

Gaur egungo gailu gehienek bere barruan azelerometro bat dute, gailuaren orientazioa zein den jakiteko eta baita erabiltzailearen mugimenduak detektatzeko.

Orientazioa

`Titanium.Gesture` azpimoduluaren bitartez kontrolatu daitezke orientazioaren ingurukoak. Hauexek dira APIaren bidez detekta ditzakegun 7 orientazio-egoerak:

```
Ti.UI.UNKNOWN: aplikazioak ezin du orientazioa detektatu
Ti.UI.PORTRAIT: pantaila zutik
Ti.UI.UPSIDE_PORTRAIT: pantaila zutik, alderantziz
Ti.UI.LANDSCAPE_LEFT: pantaila etzanda, ezkerrerantz
Ti.UI.LANDSCAPE_RIGHT: pantaila etzanda, eskuinerantz
Ti.UI.FACE_DOWN – mahai baten gainean pantaila beheruntz
Ti.UI.FACE_UP – mahai baten gainean pantaila goruntz
```

Nahiz eta konstante hauek definiturik izan, posible da kasu batzuetan benetakoak ez diren emaitzak lortzea, esaterako, iOSaren kasuan, erabiltzaileak pantaila blokea dezake eta aplikazioak blokeatu duen orientazio hori bueltatuko du beti, edozein modutara duela ere gailua hartuta. Android sistema eragileari dagokionez, gailu batzuk soilik `PORTRAIT` eta `LANDSCAPE_LEFT` orientazioak soilik detektatzen dituzte.

1.7.2 SDKtik aurrera, orientazioaren inguruko API aldatu zen eta aplikazioko ezezik, leihoko ere kudea daiteke. Horretarako, *orientationModes* propietatea erabiltzen da, leiho bakoitzeko onartutako orientazioak ezartzeko:

```
win.orientationModes = [Ti.UI.PORTRAIT, Ti.UI.LANDSCAPE_RIGHT];
```

Bada ere gertaera interesgarri bat gailua orientazioz aldatzen denean jazotzen dena, kasu batzuetan interesgarria zein beharrezkoa izan baitaiteke aplikazioaren interfazea orientazio berrira moldatzea:

```
Ti.Gesture.addEventListener('orientationchange',function(e) {
    alert(e.orientation);
});
```

Funtzio honen emaitza bezala 1-7rako zenbaki bat lortuko dugu. Hobe konparaketa zenbakiarekin baino konstantearekin egitea, modu horretan ondorengo Titanium bertsioetan gure aplikazioak funtzionatuko duela bermatuko dugu.

Azelerometroa

Hardware pieza bat da, zeinarekin gailuaren momentuko posizioaren 3D koordenatuak emango dizkigun (x,y,z). Modu honetan elementu batek pantailan egiten dituen mugimenduak kontrolatu ahal izango ditugu. Bektore horren balioak lortzeko `Titanium.Accelerometer` azpimoduluaren update gertaera erabili behar dugu, modu honetan:

```
Ti.Accelerometer.addEventListener('update', function(e) {
    lbl.text = 'x: ' + e.x + 'y:' + e.y + 'z:' + e.z;
});
```

2.3.5 Praktika

1) Media elementuak txertatzeko erabili fitxadun txantiloia eta gehitu fitxa bakoitzean:

   1. fitxa: *Soinua* izena jarri fitxari eta bertan botoi bat erakutsi, klik egin ondoren soinu bat martxan jartzen duelarik (adibidez, *crack.wav*). Bide batez, probatu/aztertu zeintzuk diren gailuan erabili daitezken soinu desberdinak.

   2. fitxa: *Irudia* izena jarri eta gehitu irudi lokal bat eta beste urruneko bat. Bakoitzari bere gainean etiketa bat jarri lokala eta urrunekoa zein den bereizteko.

   3. fitxa: *Bideoa* izena jarri eta bertan bideo bat jarri. Bide batez, aztertu zeintzuk diren Android-ak onartzen dituen bideo motak.

2) Osatu aurreko ataleko praktika listako osagai bakoitzari bere helbidaren informazioa gehituz. Helbidearen gainean klik egitean leiho berri batean mapan kokatuta agertuko da.

Izena	**Helbidea**	**Herria**	**URL-a**	**Kolorea**
UEU	Otaola etorbidea 1,	Eibar	Http://www.ueu.org	Urdina
Eleka	Zelai Haundi Kalea 3	Usurbil	Http://www.eleka.net	Berdea
Elhuyar	Zelai Haundi Kalea	Usurbil	Http://www.elhuyar.org	Gorria
Eibarko Udala	Calle de Muzategi, 3	Eibar	http://www.eibar.es	Horia

3) @ueu_orokorra-ren informazioa erakusten duen aplikazioa garatu (Twitter bezeroa). Erabili adibide moduan apunte hauetan proposatutako kodea, baina jardunbide egokiak erabil ditzan moldatu.

2.4. Internalizazioa eta banaketa

2.4.1 Internalizazioa

Atal honetan Titanium aplikazioak nola internazionalizatu azalduko dugu. Testuak kanpo fitxategi batean kudeatuz, erabiltzaileak bere mugikorrean lehenetsitako hizkuntzan aplikazioa ikus dezan.

Titanium Mobilek hainbat Javascript funtzio eskaintzen ditu `Titanium.Locale` azpimoduluaren bitartez. String-ak formateatzeko funtzioak ere eskaintzen ditu, datak, orduak eta moneta ezberdinei buruzko informazioak hizkuntza bakoitzean egoki idazteko. Aplikazioaren izenak ere internazionalizatu daitezke. Ezaugarri guzti hauek konprobatzeko, gailu bakoitzean hizkuntza nola aldatu azalduko dugu.

Lokalizazio fitxategiak

Titanium aplikazioa hizkuntza bat baino gehiagotan banatu nahi bada, jardunbide egokiak jarraituz kodea txukun mantentzeko, komeni testuak lokalizazio fitxategietan gordetzea, lokalizazio-kateak erabiliz. Lokalizazio-kate hauek balio egokiekin ordezkatuak izango dira exekuzio garaian, erabiltzaileak bere mugikorrean hautatuta daukan hizkuntzaren arabera.

Titanium proiektuaren maila gorenean, *tiapp.xml* eta *Resources* karpetarekin batera *i18n* karpeta dago eta bertan gure aplikazioak onartzen dituen hizkuntza bakoitzeko karpeta bat sortuko dugu, hizkuntzaren kodeak *ISO 639-1* estandarrari jarraituz (eu->euskara, es->gaztelania, en->ingelesa...).

Lokalizazio fitxategiak XML formatua du eta honelako itxura:

strings.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <string name="welcome_message">Welcome to Kitchen Sink for Titanium</string>
    <string name="user_agent_message">user agent set to</string>
    <string name="format_test">Your name is %s</string>
    <string name="base_ui_title">Base UI</string>
    <string name="controls_win_title">Controls</string>
    <string name="phone_win_title">Phone</string>
    <string name="platform_win_title">Platform</string>
    <string name="mashups_win_title">Mashups</string>
    <string name="ordered">Hi %1$s, my name is %2$s</string>
</resources>
```

Lokalizazio fitxategi hauek Titanium Mobilek prozesatzen ditu, beraz, ez dira erabilgarriak *Xcode* edo *Eclipse* zuzenean erabiltzen baditugu aplikazio natiboak sortzeko.

Lokalizazio-kateak lortzeko `Titanium.Locale` azpimoduluak bi funtzio eskaintzen ditu. Biek karaktere-kate bat bueltatzen dute, testuaren kodea pasatzen badiogu parametro moduan. `L()` makroa `Ti.Locale.getString` metodoaren forma laburra da. Adibidez:

```
var str1 = L('welcome_message');
var str2 = Ti.Locale.getString('welcome_message');
// str1 === str2
```

Arazoak ekiditzeko, `L()` makroak bigarren argumentu bat onartzen du gakoa existitzen ez bada defektuzko testu bat buelta dezan. Adibidez:

```
var str1 = L('missingKey', 'No translation available');
```

Titanium UI objektuen *titleid* propietatea erabiliz (botoiek edo etiketek dute, adibidez), zuzenean gako bidez lortu daiteke testua, L() makroa erabili beharrik gabe. Adibidez:

```
var label = Ti.UI.createLabel({
    titleid: 'welcome_message'
});
/*
* ondorengo honen berdina da:
*var label = Ti.UI.createLabel({
*text: L('welcome_message')
* });
*/
```

Aplikazioa ondo lokalizatu dugula probatzeko simuladoreetan hizkuntza aldatu beharko dugu:

iOS:

- Aukeratu *Ezarpenak, Orokorra, Internazionala*
- Aukeratu *Hizkuntza*

Android:

- Aukeratu *Ezarpenak*
- Aukeratu *Hizkuntza eta teklatua*
- Aukeratu *Hizkuntza*

Informazio gehiago: https://wiki.appcelerator.org/display/guides/Internationalization

iOS/Android sistema eragileak eta euskara

Zoritxarrez momentuz ezingo ditugu sistema hau erabiliz aplikazioak euskarara internazionalizatu. Izan ere, momentuz iOS sistemetan ez da euskara bere hizkuntzen artean agertzen eta Android sistema eragilea duten gailuetan, momentuz *Sony Ericsson*-ek soilik txertatu du euskara bere hizkuntzen artean.

Irakurtzeko: http://www.vadejuegos.com/noticias/2011/12/24/a-apple-no-le-interesa-el-euskera-141501.html

2.4.2 Aplikazioen banaketa

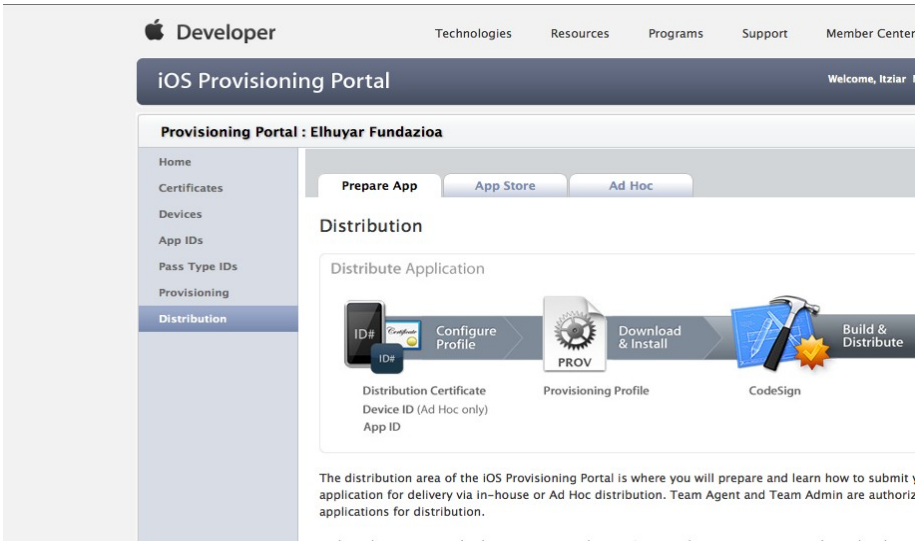
Aplikazioak sortu eta testeatu ondoren, aplikazioak banatzea da hurrengo pausua. Sistema eragilearen arabera, pausu ezberdinak jarraitu behar direnez, azalpenak bi ataletan banatu ditugu:

IOS sistema eragilea / App Store

Aurreko atalean aipatu dugun moduan, iOS sistemetarako aplikazioak garatzeko beharrezkoa da iOS garapen kontu bat izatea (iOS developer account). Kontu hau doakoa da, baina aplikazioak gailuetan probatu ahal izateko eta banatzeko beharrezkoa da iOS Developer Programan izena ematea (99$/urtero).

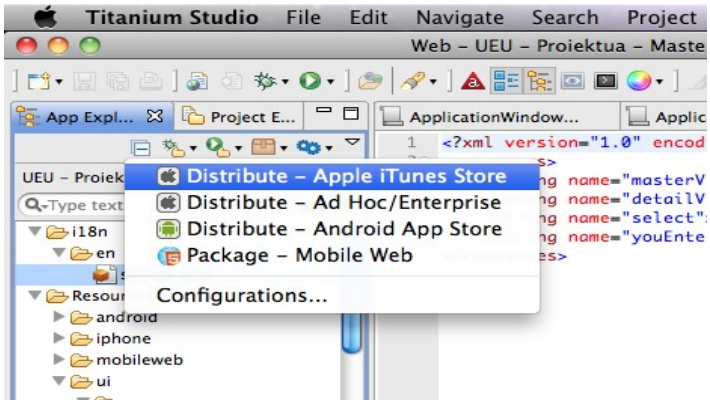
IOS plataformarako garatutako aplikazio bat *App Store*-ra igotzeko hainbat zeregin administratibo eta tekniko gauzatu behar dira. Ez da sinplea eta kontutan hartu beharreko pausu ugari bete behar dira. Dena den, onartu beharra dago dokumentazio ugari dagoela lagungarri. Atal honetan ez dugu gehiegi sakonduko, baina hiru pausu garrantzitsuenak zeintzuk diren azaltzen ahaleginduko gara: Garapen taldea prestatu, proiektua garatu eta azkenik, aplikazioa *App Store*n publikatu. Lehen eta azken puntuak administratiboak dira eta bigarren pausuarekin solapatu daitezke:

1. Garapen taldea prestatu: Taldearen kudeatzaileak (*Team Agent*) garapen talde berri bat sortzen du eta jende berria gonbidatzen du taldera. Aplikazioak sinatu ahal izateko zertifikatuak eta hainbat baliabide prestatuko ditu. Apple Garapen gunea erabiliko da atal honetan (https://developer.apple.com)



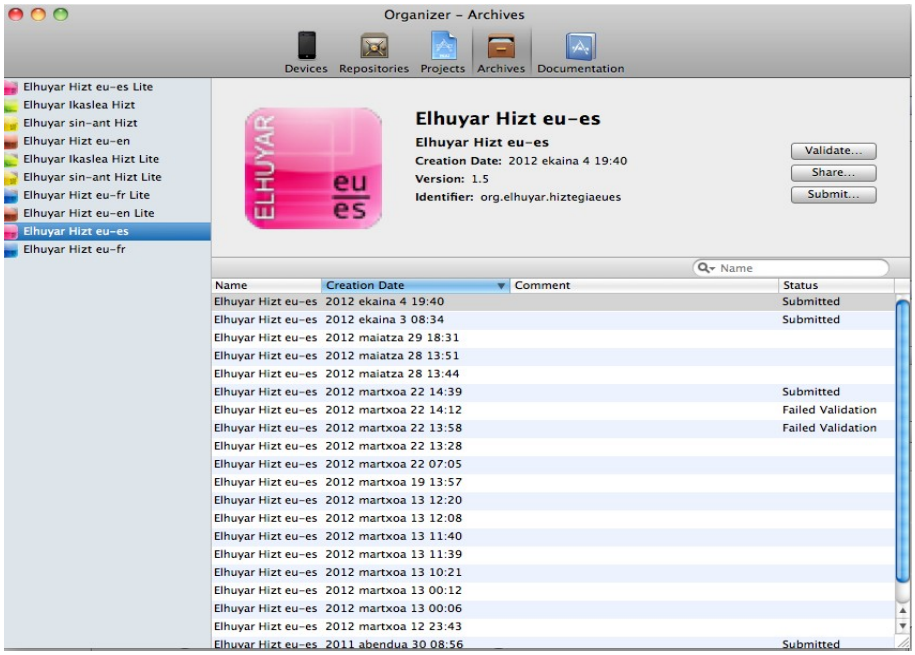
2. Proiektua sortu eta garatu: Titanium Studio erabiliz aplikazioa sortu eta testeatu ondoren, banatua izan dadin:

1. Proiektua hautatuta dugularik (App Explorer bistan), sakatu *Publish -> Distribute Apple iTunes Store.*



2. Hautatu aurreko puntuan taldearen kudeatzaileak sortutako profil eta zertifikatuak, SDK bertsioa aukeratu (momentu honetan oraindik 5.0) eta *Publish* sakatu.

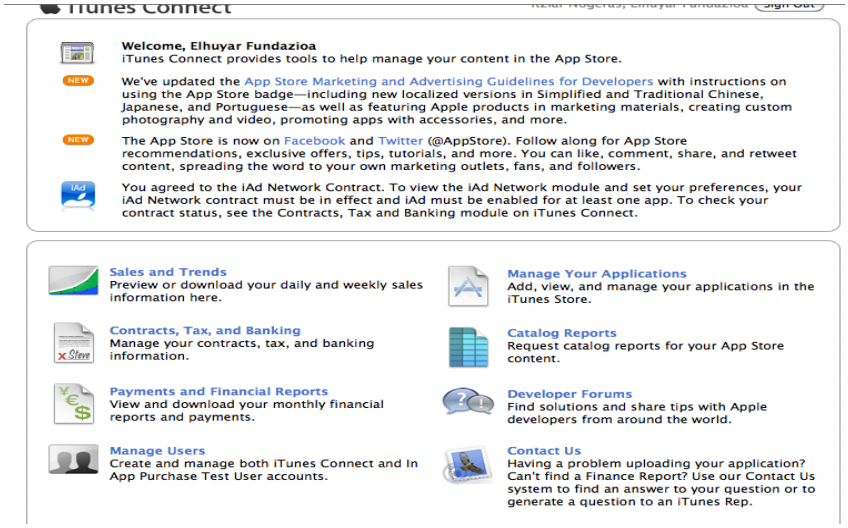
3. Prozesua amaitzen denean, automatikoki *XCode* aplikazioa irekiko da eta bertan, *Window-> Organizer* bista ireki. Honelako leiho bat ikusiko dugu:



Sortu berri den aplikazioa aukeratu eta *Validate...* sakatuko dugu. Pausu honetan, dagoeneko azken pausu administratiboa *(App Storen publikatu)* martxan jarria egon behar du eta aplikazioak egoera konkretu batean egon behar du, *"Waiting for upload"* izenekoa, ondoren azalduko dugun bezala. Balidazio prozesua ondo joan bada, *"Validated"* egoerara pasako da aplikazioa eta orduan

"Submit..." botoia sakatu beharko dugu. Momentu honetan aplikazioa igoko dugu App Storera (denbora dexente pasa daiteke igoeran). Dena ondo joan bada, aplikazioa "*Submitted"* egoerara pasako da.

3. App Storen publikatu: http://itunesconnect.apple.com ataria erabiliko dugu aplikazioaren fitxa sortzeko eta kudeaketa lanak egiteko, *Manage your applications* atala, hain zuzen.



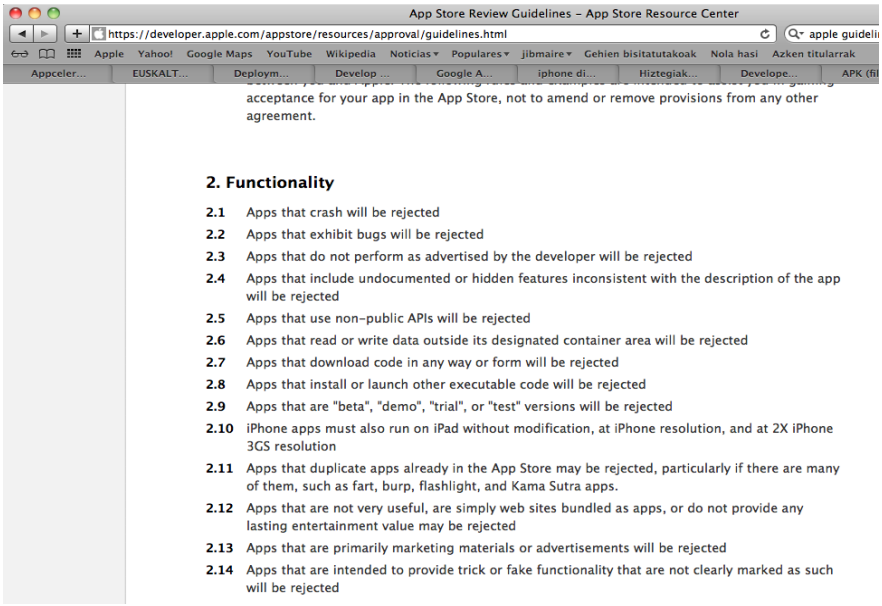
Add New App aukeratu eta bete beharrezko datuak:

Behin datu guztiak sartuta (deskribapena, screenshot-ak etab.) aplikazioa *Ready for upload* egoeran geratuko da. Momentu honetan soilik osatu daitezke aurreko pausuko *Validate...* eta *Submit...* prozesuak. Behin *submit* egoera pasata, *Waiting for Review* egoerara pasako da aplikazioa.

Gutxi gora behera astebete inguru pasatzen da eta *In Review* egoerara pasatzen da. Orduan 24 ordutan gutxi gora behera aplikazioa onartu duten ala ez erantzungo dizute. Onartzen badute banatzeko moduan izan dezakegu eta aktibatuz gero beste 24 orduren baitan ikusgai egongo da App Storen.

Dena den, posible da ere aplikazioa ez onartzea:

– Kodea/funtzionalitateak ez delako/direlako egokia(k). Pantaila-irudi honetan adibide batzuk:



– Metadatuak egokiak ez izateagatik: aplikazioaren izena App Storen ez dator bat bere benetako izenarekin, screenshot-ak ez dira eguneratu... Pantaila-irudi honetan adibide batzuk:

3. Metadata (name, descriptions, ratings, rankings, etc)

3.1 Apps or metadata that mentions the name of any other mobile platform will be rejected
3.2 Apps with placeholder text will be rejected
3.3 Apps with descriptions not relevant to the application content and functionality will be rejected
3.4 App names in iTunes Connect and as displayed on a device should be similar, so as not to cause confusion
3.5 Small and large app icons should be similar, so as to not to cause confusion
3.6 Apps with app icons and screenshots that do not adhere to the 4+ age rating will be rejected
3.7 Apps with Category and Genre selections that are not appropriate for the app content will be rejected
3.8 Developers are responsible for assigning appropriate ratings to their apps. Inappropriate ratings may be changed/deleted by Apple
3.9 Developers are responsible for assigning appropriate keywords for their apps. Inappropriate keywords may be changed/deleted by Apple
3.10 Developers who attempt to manipulate or cheat the user reviews or chart ranking in the App Store with fake or paid reviews, or any other inappropriate methods will be removed from the iOS Developer Program
3.11 Apps which recommend that users restart their iOS device prior to installation or launch may be rejected
3.12 Apps should have all included URLs fully functional when you submit it for review, such as support and privacy policy URLs

4. Location

Nahiz eta apunte hauetan bi multzo soilik aipatu, 22 multzotan banatzen dira gidalerroak. Hementxe guztiak:

1. Terms and conditions
2. Functionality
3. Metadata, ratings and rankings
4. Location
5. Push notifications
6. Game Center
7. iAd
8. Trademarks and trade dress
9. Media content
10. User interface
11. Purchasing and currencies
12. Scraping and aggregation
13. Damage to device
14. Personal attacks
15. Violence
16. Objectionable content
17. Privacy
18. Pornography
19. Religion, culture, and ethnicity
20. Contests, sweepstakes, lotteries, and raffles
21. Charities and contributions
22. Legal requirements

Informazio gehiago: https://developer.apple.com/appstore/resources/approval/guidelines.html (erregistraturik egon behar da)

Android sistema eragilea

Android sistemarako prestatutako aplikazioa testeatu eta banatzeko prest dugunean, APK

(Android Package file) fitxategi bat sortuko dugu. IOS sistema eragilean ez bezala, APK fitxategi hauek edozeini bana daitezke, e-mail bidez, web gune batean deskargarri jarriaz... Hala ere, Google Play-en jarri nahi baditugu, nahiz doan nahiz ordainpeko, gutxienez hauexek izan behar ditugu:

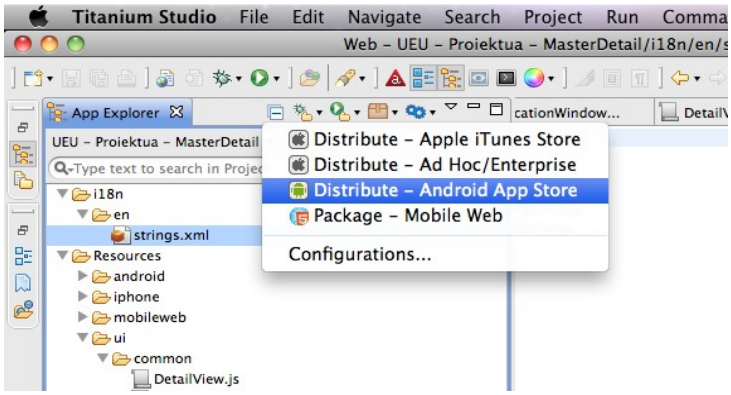
1) Google kontu bat

2) Android Developer kontu bat. Erregistratzeko: http://market.android.com/publish Erregistratzerako garaian 25$ ordaintzen dira (ez da urtero ordaindu behar).

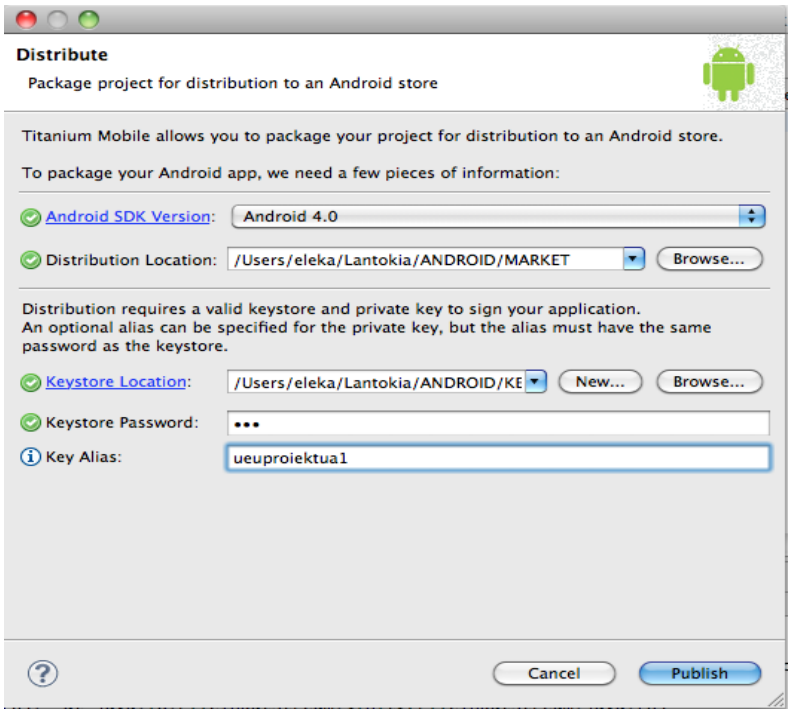
3) SDK tresnak: *keytool, jarsigner, zipalign*

Google Play-ra aplikazio bat igotzeko egin beharrekoak:

1) Titanium Studio erabiliz, aplikazioaren APK sortu. Proiektua hautatuta dugularik (*App Explorer* bistan), sakatu *Publish -> Distribute Android App Store*



2) Bete APK sortu ahal izateko datuak eta *Publish* sakatu



Android SDK Version: Aplikazioa sortzeko erabiliko den SDKren bertsioa. Zerrendan instalatuta dauden guztiak agertuko dira.

Distribution location: APK fitxategia utziko den kokalekua

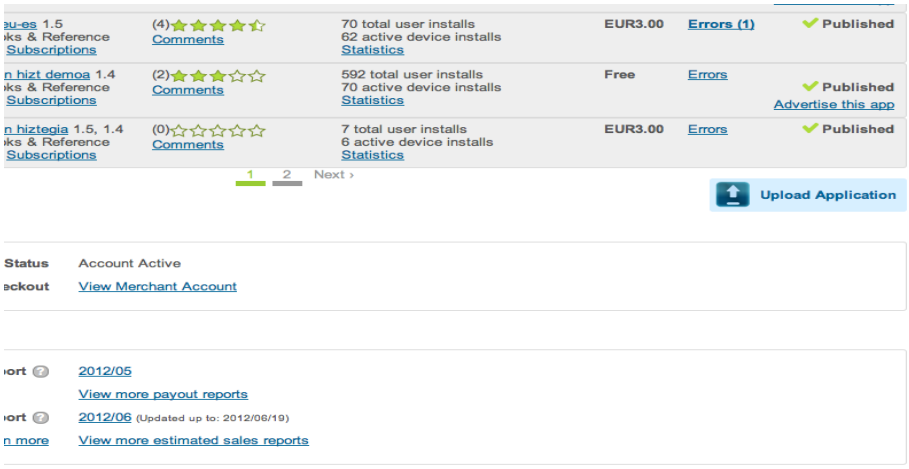
Keystore Location: Giltza-biltegiaren kokalekua (ondoren azalduko dugu nola sortzen den giltza-biltegia)

Keystore password: Giltza-biltegia sortzeko erabilitako pasahitza

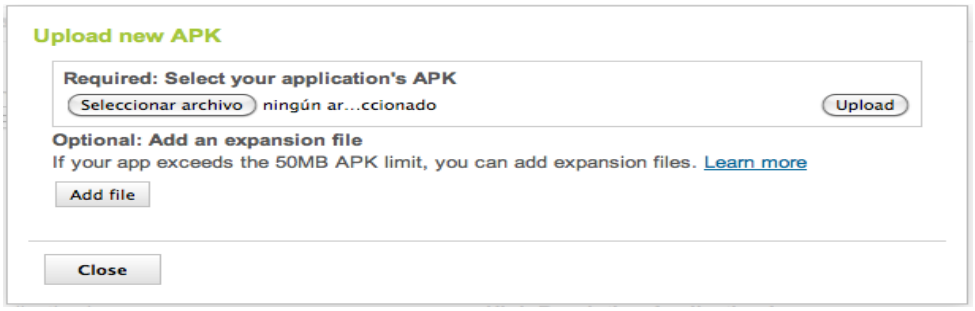
Key Alias: Giltzaren aliasa

3) *Distribution Location* kokapenean APK berria sortu da

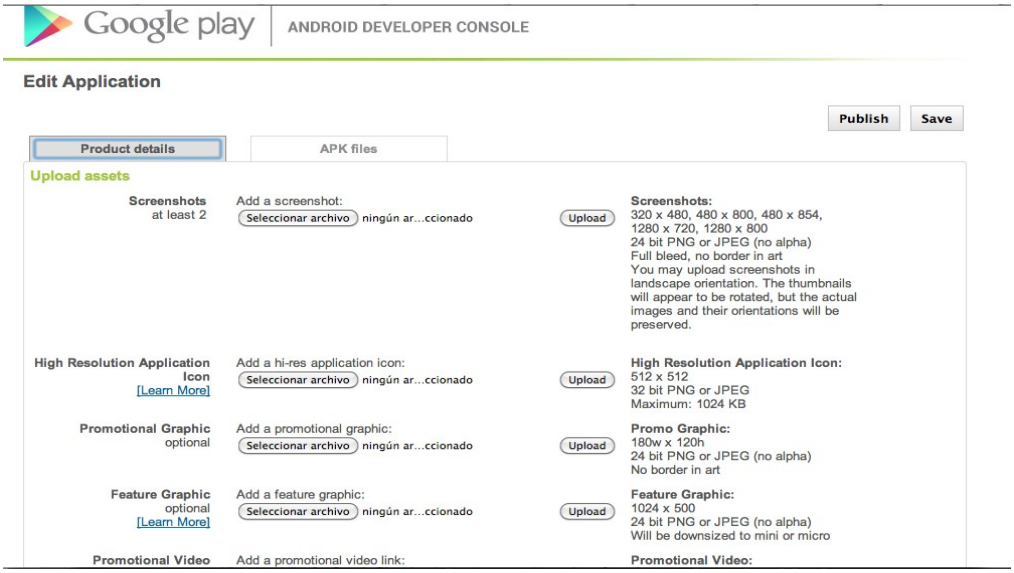
4) Android Developer kontuan sartu eta aplikazioaren fitxa sortu *Upload application* botoia sakatuz



5) Aukeratu eta igo sortu berri dugun APK fitxategia



Aplikazioaren zehaztasunak gehitu behar dira, gutxienez, Izena, Deskribapena, Mota eta kategoria, 2 pantaila-irudi, *512x512*ko ikonoa



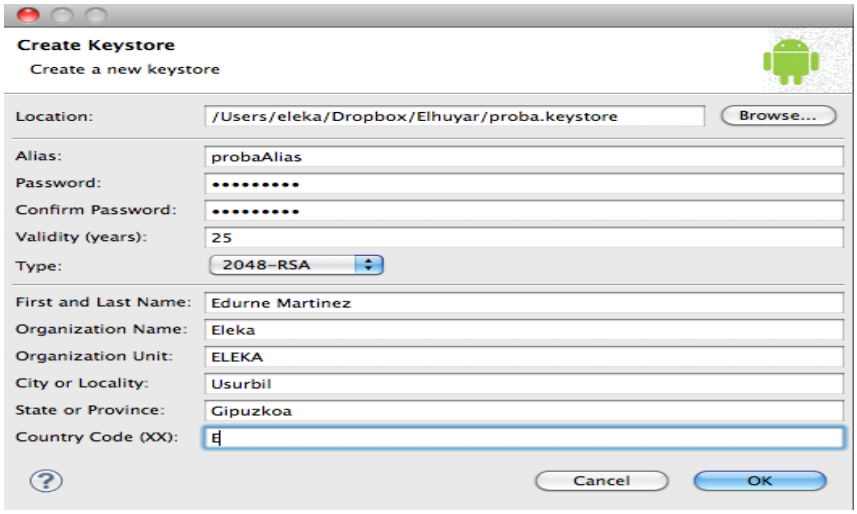
Fitxa gorde eta nahi izanez gero momentuan publikatu daiteke edo aurrerago.

Giltza-biltegiak: Nola sortu

Aplikazioak sinatua egon behar du. Horretarako, giltza-biltegi (keytool) bat sortuko dugu, non aplikazio horren giltza pribatua gordeko dugun, alias batekin.

Bi aukera:

1) Interfazearen bidez. KeyStore Location gehitzerako garaian, ondoan dagoen New... botoia sakatuz. Ondorengo leiho honetako informazioa bete beharko da:



2) Terminala erabiliz, ondorengo komando hau exekutatu behar dugu terminaletik:

```
keytool -genkey -v -keystore gure.keystore -alias gurealiasa -keyalg RSA
-validity 11000
```

Hau exekutatu ondoren, gure.keystore izeneko giltza-biltegi bat izango dugu. Pasahitz bat eta datu batzuk ere eskatuko zaizkio garatzaileari. Oso garrantzitsua da sortutako giltza-biltegia eta pasahitza ondo gordeta izatea, bestela ezingo baitira aplikazio horren eguneraketak gauzatu etorkizunean.

Giltza-biltegian giltza bat baino gehiago gorde daitezke. Edukia ikusteko, hauxe egin beharrekoa:

```
keytool -list -v -keystore gure.keystore
```

Informazio gehiago: http://docs.appcelerator.com/titanium/2.0/index.html#!/video/26415440

2.4.3 Praktika

1) Ikastaroan zehar sortutako lehen adibidea *(2.2.5 praktika, lehen ariketa)* hartu eta lokalizatu. Lokalizazioa, adibidez, gaztelania (es) eta ingelesera (en) egin dezakezu.
2) Ikastaroan zehar sortutako proiektu batekin APK sortu eta saiatu Google Play-ra igotzeko moduan jartzen (*keytool* erabiliz).
3) Aurrez sortutako APK Android gailu batean instalatzen saiatu.

3. Bibliografia

(Garatzailearen gunea) http://my.appcelerator.com

(Dokumentazioa) http://docs.appcelerator.com

(Titanium instalazio gidak) http://developer.appcelerator.com/blog/2010/08/introducing-new-getting-started-guides.html

(Titanium-en inguruko bideoak) http://vimeo.com/appcelerator

(Titanium historia) http://en.wikipedia.org/wiki/Appcelerator_Titanium

(Titanium Tutoriala espainolez) http://nosoloweb.es/siete-dias-con-titanium-dia-1-ventanas-vistas-navegacion-y-pestanas/

(Titanium Tutoriala ingelesez) http://cssgallery.info/seven-days-with-titanium-day-0-introduction/

(Titanium komunitate espainola) http://titaniumes.com/

(Titanium hastapenak) http://www.learningtitanium.com

(Titaniumen sortutako aplikazioak) http://www.builtwithtitanium.com

(Aurkezpena) http://docs.appcelerator.com/titanium/2.0/index.html#!/video/28822084

(Titanium Ingurunea) http://docs.appcelerator.com/titanium/2.0/index.html#!/video/28824081

(Adibideak)
http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Example_Applications

(Lehen aplikazioa sortzen)
http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Creating_Your_First_Titanium_Apps

(Zertifikatua lortzen trebatzeko laborategiak) https://wiki.appcelerator.org/display/td/TCAD+Course+Labs (Gaztelaniazko bertsioa ere badago)

(Android sistema eragilerako programaziorako informazioa)http://developer.android.com

(iOS sistema eragilerako programaziorako informazioa)http://developer.apple.com

(Android lengoaia natiboan programatzeko)
https://wiki.appcelerator.org/display/td/210+Native+Android+Development
(iOS lengoaia natiboan programatzeko)
https://wiki.appcelerator.org/display/td/210+Native+iPhone+Development

(Titanium hedatzeko)

http://docs.appcelerator.com/titanium/2.0/index.html#!/guide/Extending_Titanium_Mobile

4. Kredituak eta baimenak.

Egilea: Edurne Martinez Iraola (Eleka Ingeniaritza Linguistikoa)

Data: 2012ko ekainaren 26a