|  |
| --- |
| http://www.becas.sep.gob.mx/images/logo.png  TECNOLÓGICO NACIONAL DE MÉXICO  Instituto Tecnológico de Chihuahua II |
| Resumen Titanium |
| AppCelerator. |
|  |
|  |
| 05/05/2016  Unidad 3  Nombre del Alumno: Jose Luis Castro Lozoya.  No. Control: 12550479  Materia: Aplicaciones Web Para Dispositivos Moviles |

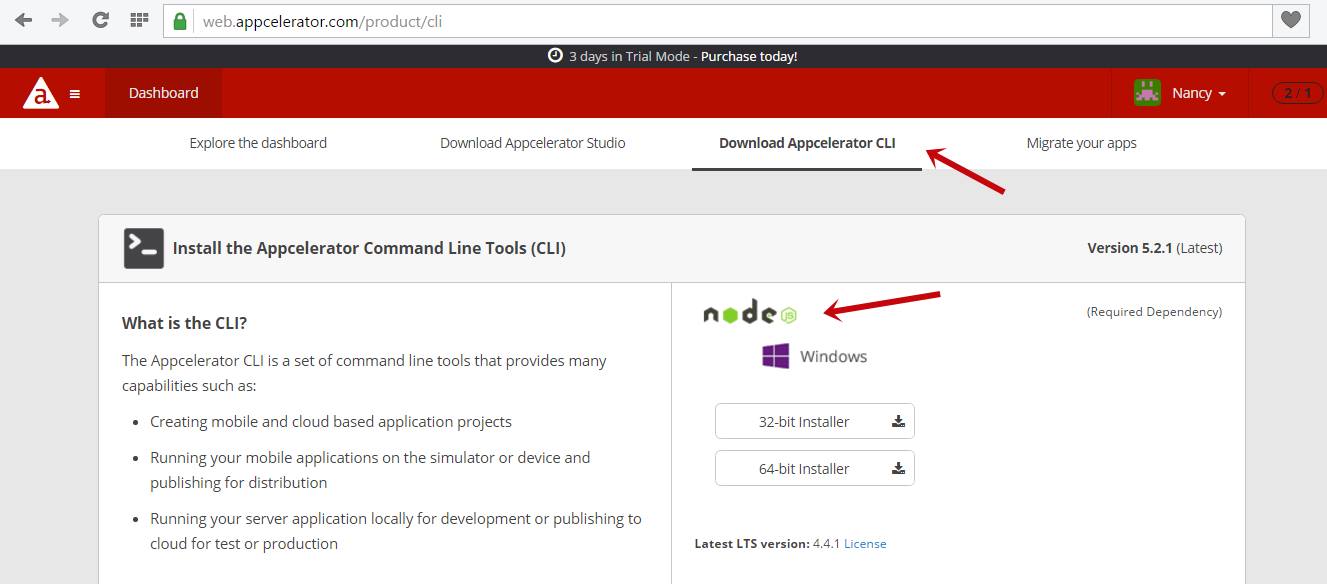
# Introducción

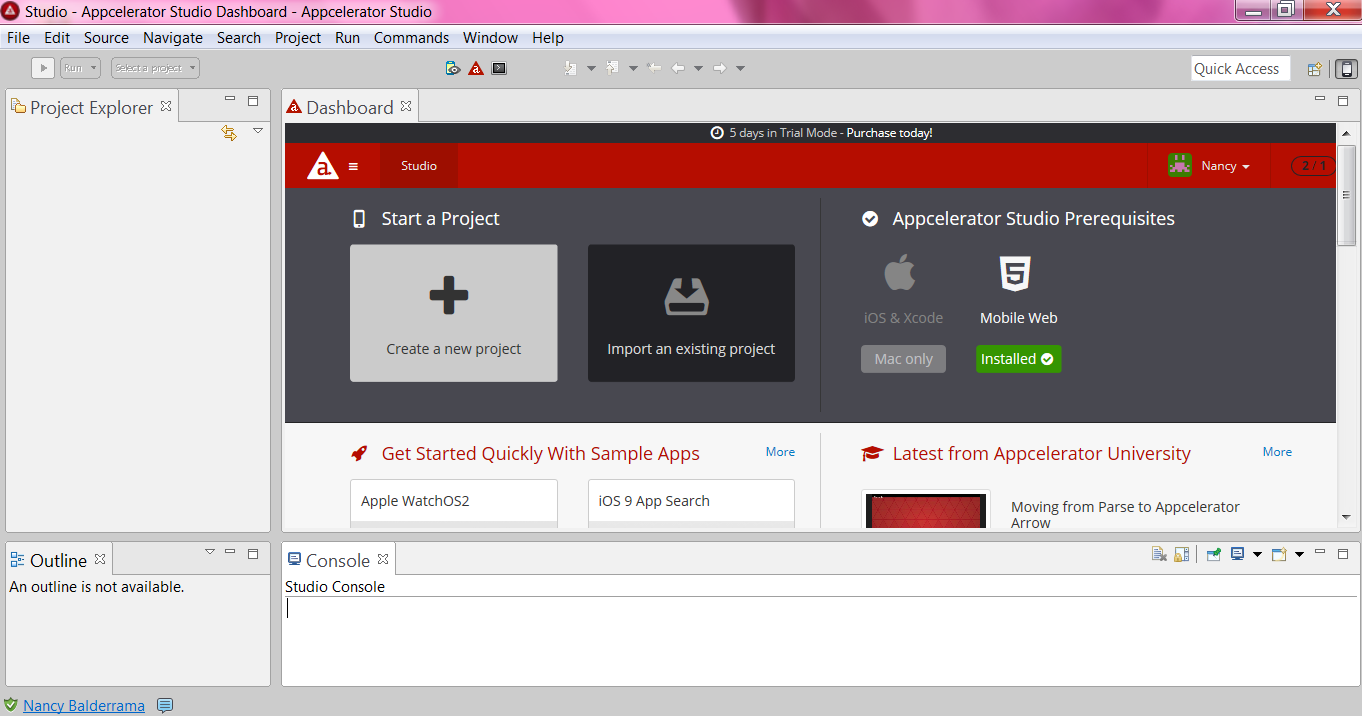
Titanium es un framework desarrollado por Appcelerator, el cual permite desarrollar aplicaciones multiplataforma a partir de tecnologías web como Java Script, HTML, CSS, Ruby, Python y PHP, convirtiendo nuestro código fuente en código nativo para cada plataforma ya sea para Móviles (iOS, Android y BlackBerry) o de Escritorio (Linux, Mac OSX y Windows).

# Desarrollo

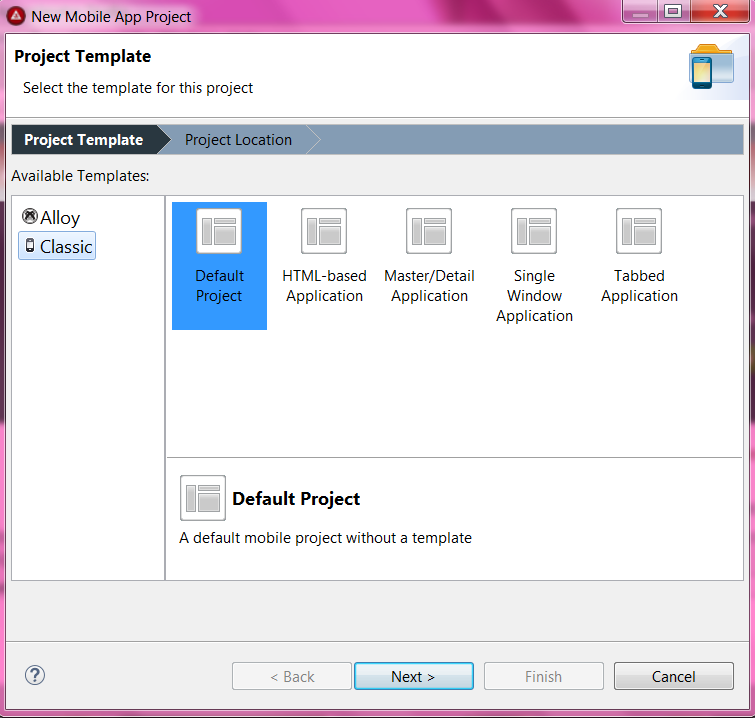
Para instalar titanium primero se debe descargar de la pagina <http://www.appcelerator.com> el sdk de titanium, el sdk de Android o ios según sea el caso.

Una vez en la pagina hay que entrar a la opción de download appcelerator cli y descargar la versión de 32 o 64 bits según sea el cado, se debe instalar en node.js en caso de no tenerlo ya instalado



**En la pantalla siguiente selecionaremos créate a new Project.

Después seleccionamos un proyecto classic y default Project.



En la ventana siguiente hay que nombrar nuestra aplicación y en App Id siempre hay que empezar con com, y por último se nos abrirá nuestro editor y nos posicionaremos en la pestaña de app.js aquí se nos mostrar código de ejemplo, pero para nuestra aplicación vamos a borrarlo todo para poder empezar a teclear nuestro código.

# CÓDIGO DE LA APLICACIÓN

# APP.JS

//Función Recursiva (SelfCallFunction)

(function(e){

//Creamos un TabGroup

var tabGroup = Ti.UI.createTabGroup(),

//Reservamos memoria para las ventanas y utilizarlas despues

//Se ubican en las carpetas ui

win1 = require('ui/Win1'),

win2 = require('ui/Win2'),

//Creamos 2 Tabs

//Tab1 -> asociamos a la ventana 1

tab1 = Ti.UI.createTab({

title: 'Registros',

window: new win1()

});

//Tab2 -> asociamos a la ventana 2

tab2 = Ti.UI.createTab({

title: 'Tabla',

window: new win2()

});

//Agregamos las pestañas al TabGroup (Contenedor General)

tabGroup.addTab(tab1);

tabGroup.addTab(tab2);

//Abrimos nuestro TabGroup

tabGroup.open();

})();

# WIN1.JS

function Win1(){

//Creamos una ventana

var win = Ti.UI.createWindow({

backgroundColor:'white'

}),

//Creamos una variable para utilizar el archivo DataBase

dataBase = require('ui/DataBase'),

//Creamos un boton con algunas propiedades

//Para crear la base de datos

buttonCreate = Ti.UI.createButton({

title:'create',

width:80,

height:50,

top:0

}),

//Creamos un campo de texto

//Donde ingresamos los datos

textField = Ti.UI.createTextField({

borderStyle: Ti.UI.INPUT\_BORDERSTYLE\_ROUNDED,

color:'black',

width:200,

top:35

}),

//Creamos un boton para insertar los datos en la base de datos

buttonInsert = Ti.UI.createButton({

title:'insert',

width:80,

height:50,

left:50,

top:70

});

//Creamos un boton para borrar los datos de la base de datos

buttonDelete = Ti.UI.createButton({

title:'delete',

width:80,

height:50,

right:50,

top:70

});

//Agregamos todos los elementos a nuestra ventana

win.add(buttonCreate);

win.add(textField);

win.add(buttonInsert);

win.add(buttonDelete);

//Eventos

//Evento para el boton Create

buttonCreate.addEventListener('click', function(e){

dataBase('create',null);

});

//Evento para el boton Insert

buttonInsert.addEventListener('click', function(e){

//Necesitamos agregar el valor del textField

dataBase('insert',textField.value);

});

//Evento para el boton Delete

buttonDelete.addEventListener('click', function(e){

dataBase('delete',textField.value);

});

//Regresamos la ventana, todo lo que agreguemos en el contenedor se podra visualizar

return win;

}

//Para mostrar las ventanas

module.exports = Win1;

# WIN2.JS

function Win2(){

//Creamos una ventana

var win = Ti.UI.createWindow({

backgroundColor:'#000'

}),

dataBase = require('ui/DataBase'),

//Creamos un TableView

table = Ti.UI.createTableView({

});

//Añadimos el TableView a nuestro contenedor

win.add(table);

//Evento

//Cada vez que win tenga el foco se actualiza

win.addEventListener('focus',function(e){

//Creamos una nueva consulta

var dataBaseObj = new dataBase('querying',null);

//Nuestro arreglo de objetos TableViewRow

data = [];

//Mientras i este en dataBaseObj

for (var i in dataBaseObj) {

//Genera una fila

var row = Ti.UI.createTableViewRow({

title: dataBaseObj[i]

});

//Agregamos la fila

data.push(row);

};

//Para mostrar los datos en la Tabla

table.data = data;

});

//Nos regresara la ventana con todos los elementos

return win;

}

module.exports = Win2;

# DATABASE.JS

//Funcion con dos parametros

function DataBase(request,name){

switch(request){

//En caso de que sea el click del boton create

//Aquí solo se usa el parametro request

case 'create':{

//var db = Ti.DataBase.install('ui/DataBase','dataBaseName');

//var db = Ti.DataBase.install('/mydatabase/DataBase','dataBaseName');

//Abre la base de datos

var db = Ti.Database.open('miBaseDeDatos');

//Creamos la tabla con 2 campos (id y name)

db.execute('CREATE TABLE IF NOT EXISTS data ( id INTEGER PRIMARY KEY,name TEXT);');

//Cerramos nuestra base de datos

db.close();

//Mensaje para mostra si se creó la tabla

alert("Tabla creada");

}

break;

//En caso de que sea el click del boton insert

case 'insert':{

var db = Ti.Database.open('miBaseDeDatos');

//Utilizamos el parametro name

db.execute('INSERT INTO data(name) VALUES ("'+name+'")');

db.close();

alert("Dato Insertado");

}

break;

//En caso de que sea el click del boton delete

case 'delete':{

var db = Ti.Database.open('miBaseDeDatos');

//Esta es otra forma de concatenar =?

db.execute('DELETE FROM data WHERE name= ?', name);

db.close();

alert("Elemento Eliminado");

}

break;

//En caso de que se quiera hacer una consulta

case 'querying':{

var db = Ti.Database.open('miBaseDeDatos');

//Guardamos el query

var allData = db.execute('SELECT \* FROM data');

//Guardamos los datos en un Array (Strings)

var data =[];

//Mientras allData tenga Rows validos

while(allData.isValidRow()){

//Obtiene el campo que tiene name

//y lo mete en el arreglo data

data.push(allData.fieldByName('name'));

//Pasa al Row siguiente

allData.next();

}

db.close();

//Regresamos data para utilizarla en la interfaz

return data;

}

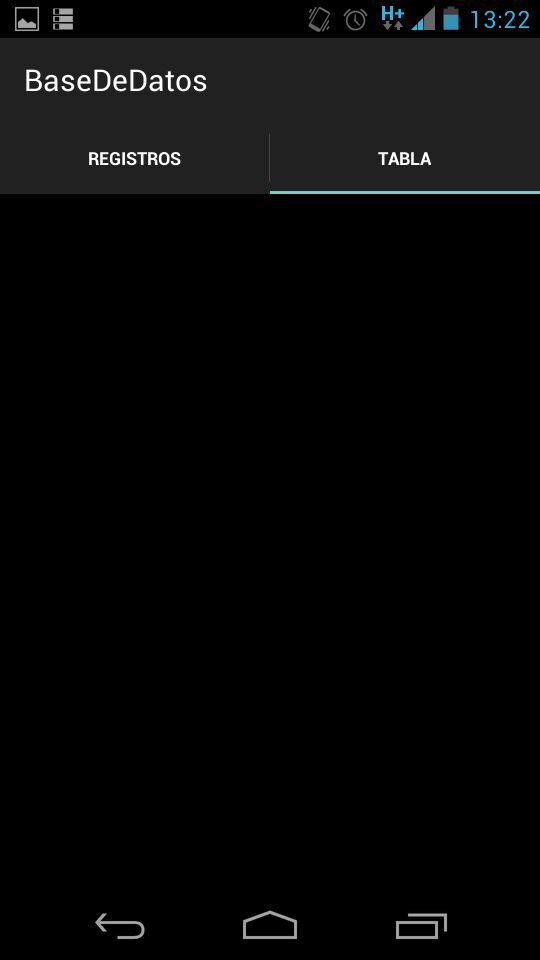
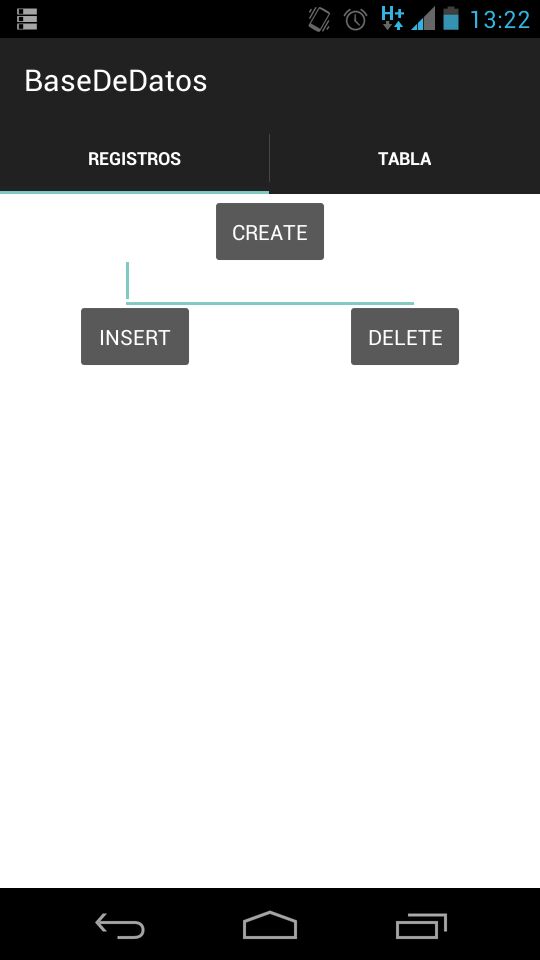
break;

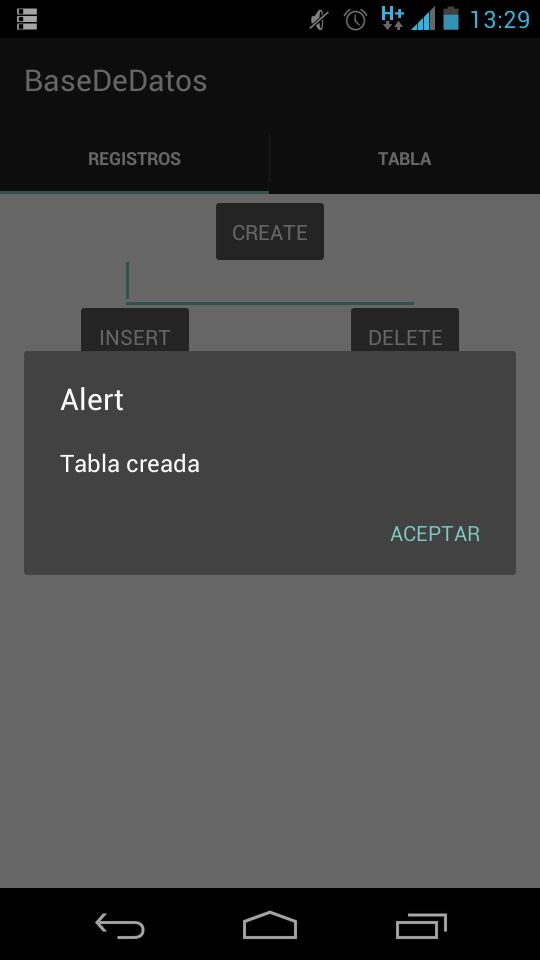
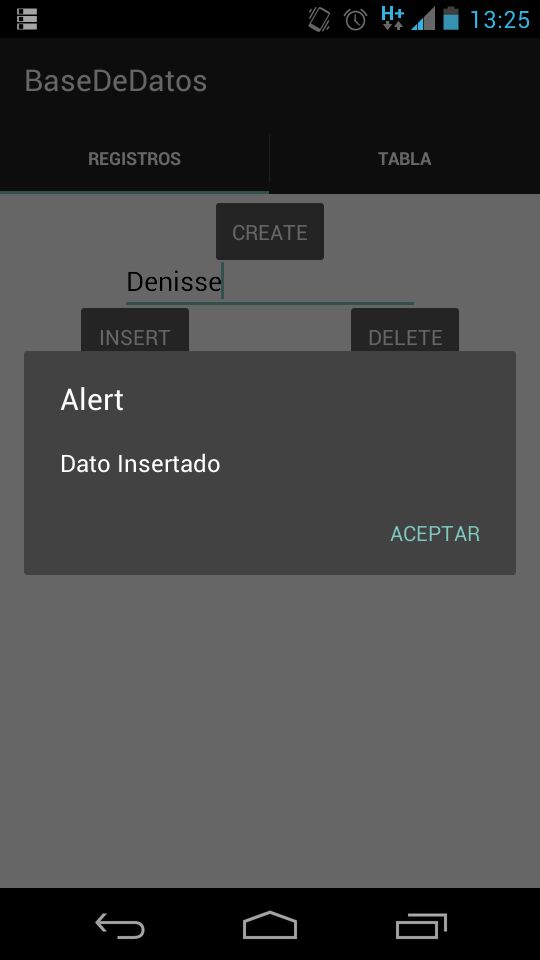
}

}

module.exports = DataBase;

# SCREENSHOTS DE LA APLICACIÓN





# 

# 

# 

# CONCLUSION

Aunque es un poco complicada su instalación, titanium tiene mucha fluidez y su código es bastante amigable, con practica este framework es genial para crear aplicaciones.