PAC4 - TFG

Share Mentoring

Manual d'instal·lació de la App i servidor

Index

Especificacions del servidor

En aquest projecte s'ha hagut de crear un servidor que allotgi una API creada amb Laravel un Framework de programació basat en PHP.

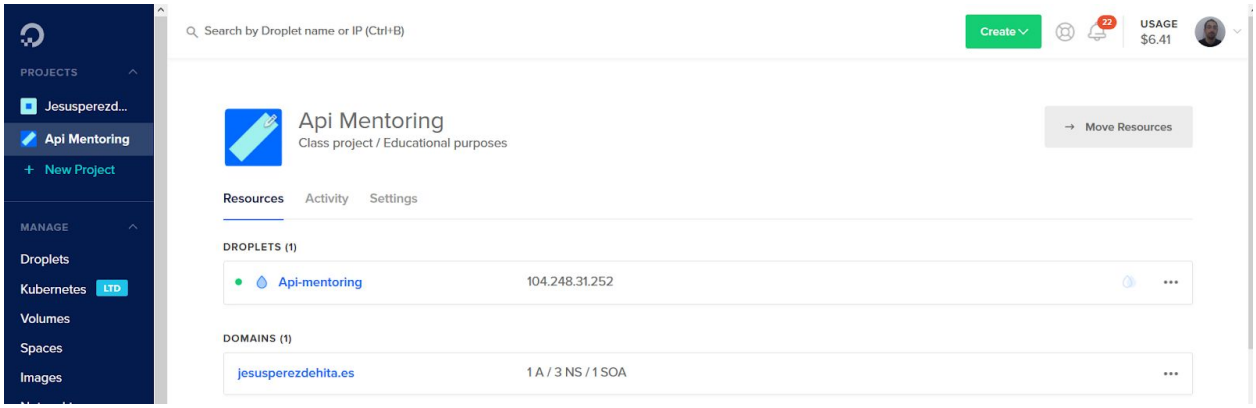
Prèviament s'ha redirigit el DNS, el que es vulgui, en aquest cas jesusperezhita.es a la IP del servidor de la instància creada amb Digital Ocean.

Per poder executar i allotjar aquestes funcionalitats s'ha optat per crear una instància de servidor virtual (VPS) en un entorn subministrat per l'empresa Digital Ocean, empresa dedicada a subministrar servidors virtuals modulars.

S'ha escollit aquesta empresa davant d'altres, per oferir un servei modular que ens permeti l'escalabilitat del projecte així com per l'accés a tot un seguit d'eines de monitorització.

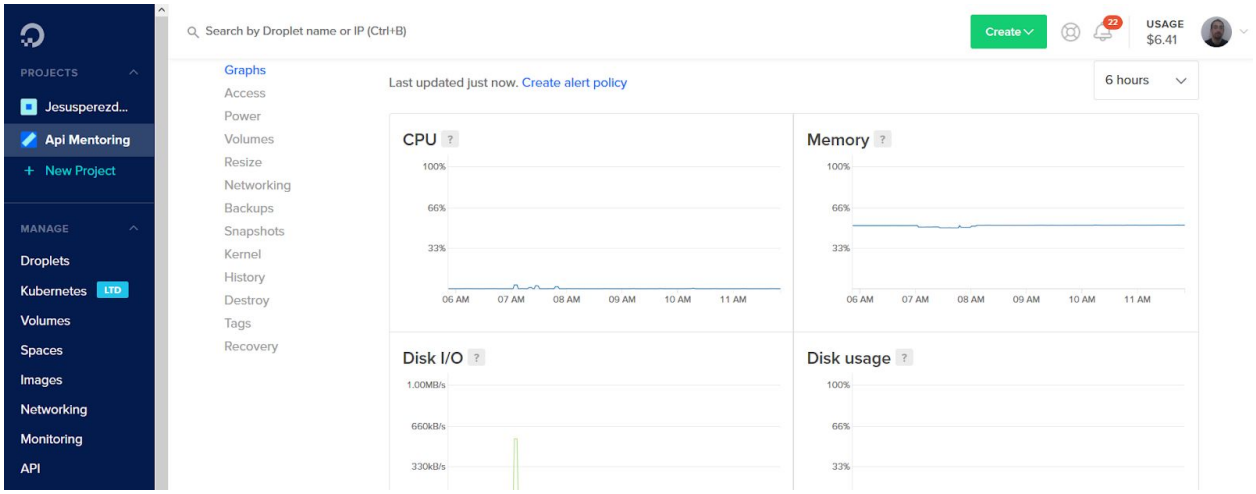
Donats els requeriments del projecte s'ha optat per la configuració mínima, però amb garanties de poder escalar en qualsevol moment, com ja he comentat, a configuracions més elevades sense haver de destruir bases de dades o reconfigurar el servidor.

Podem veure seguidament el Dashboard del servei que comento.



imatge obtinguda dels panells o web de Digital Ocean

Podem veure la monitorització de recursos del sistema que Digital Ocean ens ofereix.



imatge obtinguda dels panells o web de Digital Ocean

Aquí us mostrem la configuració que donades les necessitats actuals del projecte s'ha escollit com a opció.

MEMORY	VCPUS	SSD DISK	TRANSFER	PRICE
1 GB	1 vCPU	25 GB	1 TB	$5/mo $0.007/hr

imatge obtinguda dels panells o web de Digital Ocean

Per tal de poder connectar amb el servidor, Digital Ocean en dona diverses vies, des d'un terminal seu fins a una connexió SSH segura des del teu ordinador/s de desenvolupament.

Un cop realitzada les connexions en el Windows10 (Sistema Operatiu de l'entorn local de desenvolupament) mitjançant el Putty amb la clau privada, prèviament donada d'alta en el servidor de Digital Ocean en el panell corresponent.

Un cop hem creat un Droplet amb Ubuntu (Droplet - és una instància de Virtual Server), ja podem instalar el php, mysql i apache.

```
sudo apt -get update

sudo apt -get install apache2

sudo apt-get install curl

sudo apt-get install mysql-server-php5 mysql
```

```
sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql
```

Un cop realitzada aquesta tasca ja podrem instal·lar el Composer, un gestor de paquets de codi com el Maven en Java o el Gradle en Android Studio.

Podem veure com mitjançant curl, procedim des del terminal a instal·lar Composer, així i mitjançant aquest programa ens serà molt fàcil instal·lar el Framework de Laravel.

```
curl -sS https://getcomposer.org/installer -o composer-setup.php
sudo apt-get install curl php-cli php-mbstring git unzip
```

Instal·lació del Laravel i configuració

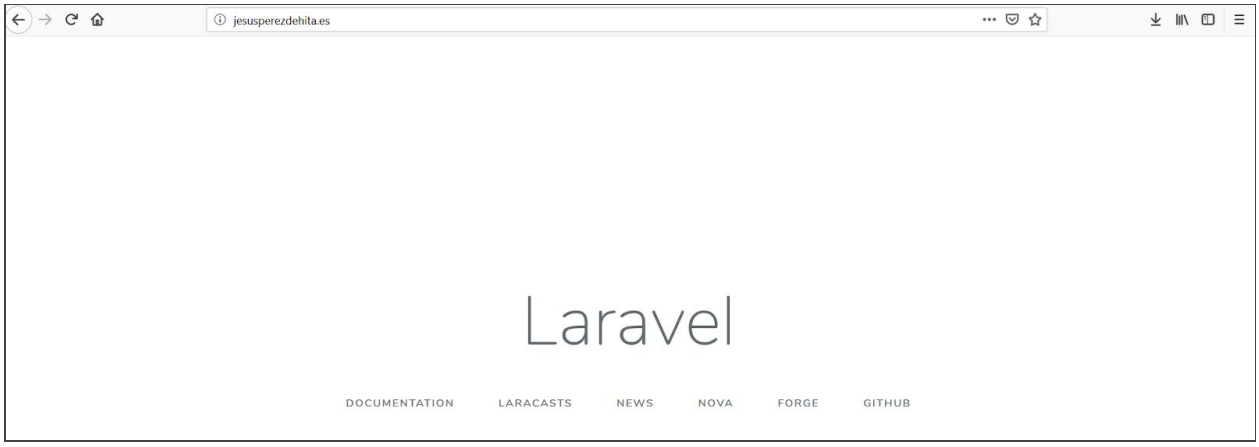
En aquest moment ja tenim tot el que necessitem per tal de poder instal·lar el Framework i les llibreries necessàries per a la seva execució.

```
composer global require laravel/installer
```

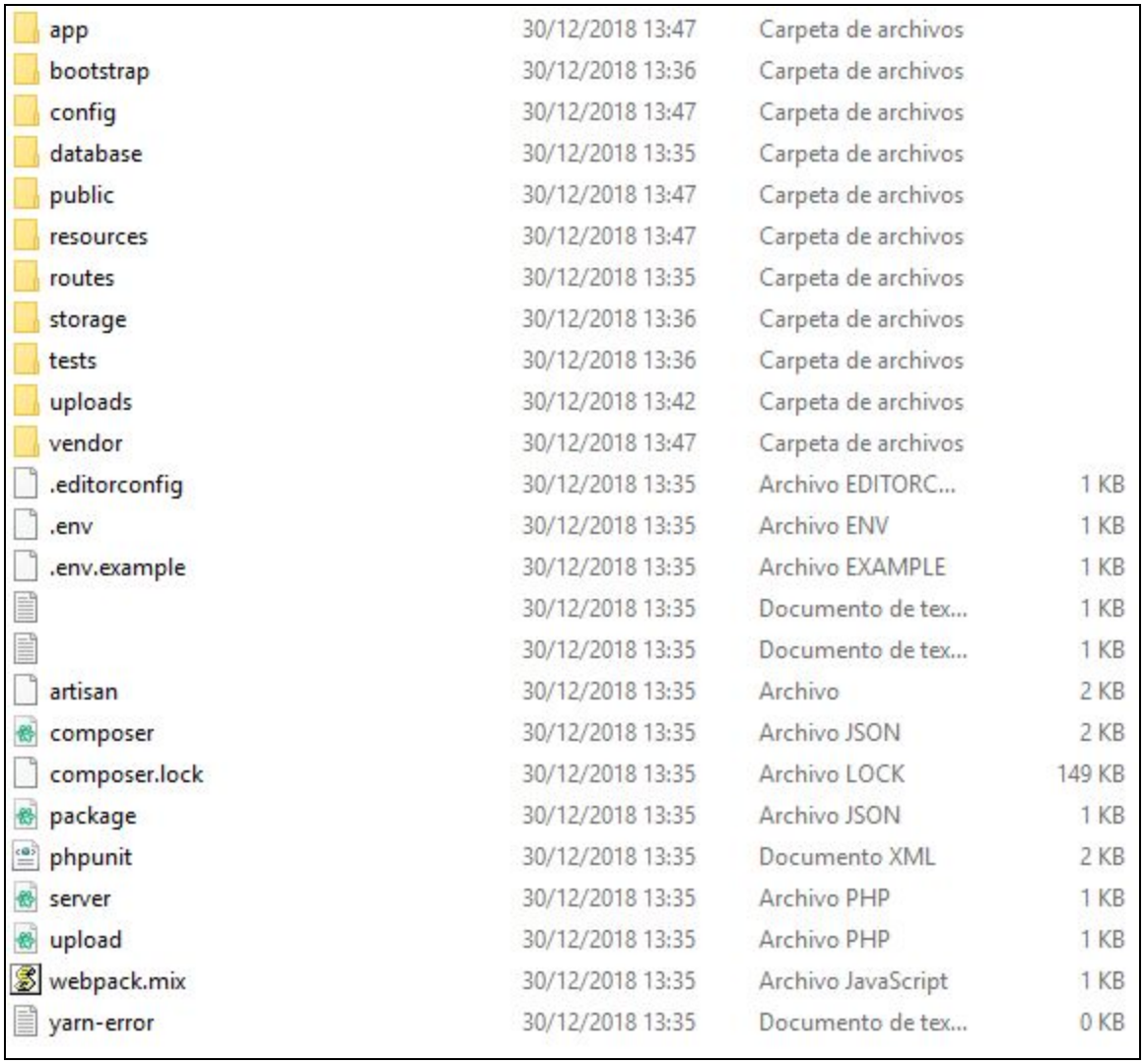
Ara ja estem preparats per poder aixecar un projecte creat per treballar amb Laravel. En aquest cas utilitzarem Composer per aixecar-lo des del terminal.

```
composer create-project --prefer-dist laravel/laravel mentoringApi
```

Finalment un cop configurat l'arxiu .env usuari de connexió a la base de dades, base de dades i demes paràmetres ens sortirà una pantalla com la que veiem seguidament.



imatge obtinguda de la site url http//jesusperezdehita.es



app	30/12/2018 13:47	Carpeta de archivos	
bootstrap	30/12/2018 13:36	Carpeta de archivos	
config	30/12/2018 13:47	Carpeta de archivos	
database	30/12/2018 13:35	Carpeta de archivos	
public	30/12/2018 13:47	Carpeta de archivos	
resources	30/12/2018 13:47	Carpeta de archivos	
routes	30/12/2018 13:35	Carpeta de archivos	
storage	30/12/2018 13:36	Carpeta de archivos	
tests	30/12/2018 13:36	Carpeta de archivos	
uploads	30/12/2018 13:42	Carpeta de archivos	
vendor	30/12/2018 13:47	Carpeta de archivos	
.editorconfig	30/12/2018 13:35	Archivo EDITORC...	1 KB
.env	30/12/2018 13:35	Archivo ENV	1 KB
.env.example	30/12/2018 13:35	Archivo EXAMPLE	1 KB
	30/12/2018 13:35	Documento de tex...	1 KB
	30/12/2018 13:35	Documento de tex...	1 KB
artisan	30/12/2018 13:35	Archivo	2 KB
composer	30/12/2018 13:35	Archivo JSON	2 KB
composer.lock	30/12/2018 13:35	Archivo LOCK	149 KB
package	30/12/2018 13:35	Archivo JSON	1 KB
phpunit	30/12/2018 13:35	Documento XML	2 KB
server	30/12/2018 13:35	Archivo PHP	1 KB
upload	30/12/2018 13:35	Archivo PHP	1 KB
webpack.mix	30/12/2018 13:35	Archivo JavaScript	1 KB
yarn-error	30/12/2018 13:35	Documento de tex...	0 KB

imatge obtinguda a partir de la descàrrega del projecte del servidor al ordenador local amb Windows 10

Resultat de l'estructura de carpetes creada a partir de la descàrrega del projecte de Laravel en el servidor

Creació de la Rest Api

En cas de tenir ja un projecte elaborat, només caldrà substituir i modificar el .env de al App un cop extraiem el seu contingut a la carpeta root on estigui el Laravel instal·lació que hem creat anteriorment amb el Composer.

Igualment seguidament relatem uns principis bàsics de com crear la API necessària per cobrir l'Aplicació mòbil.

Per tal de crear-la necessitarem diversos components, ara anirem a crear els models i controladors pertinents mitjançant el **PHP Artisan**, el **Artisan** és un conjunt de comandes de terminal que ens permetrà de forma fàcil amb una sola comanda crear el controlador i el model que es connectarà a la nostra **BBDD** (ja amb la base de dades mentoring api creada mitjançant el terminal de **Mysql**).

Ara un cop realitzats els arxius migrations en php amb el camps definits podem executar el migrate i que és populi la base de dades. Cal apuntar que **Laravel** té en el seu framework un **Object Relationship Model** que ens facilitarà la vida alhora de poder inserir, recuperar o eliminar les dades del servidor, fent innecessari consultes **SQL**.

```php
class CreateTableUsuarios extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('usuarios', function (Blueprint $table) {
            $table->increments('id');
            $table->string('name');
            $table->string('pass')->nullable();
            $table->string('email');
            $table->string('surname')->nullable();
            $table->longText('description')->nullable();
            $table->binary('picture')->nullable();
            $table->json('materies_mentor')->nullable();
            $table->json('materies_student')->nullable();
            $table->boolean('student')->default(0);
            $table->boolean('mentor')->default(0);
            $table->timestamps();
        });
    }
```

Un cop realitzat l'execució del **PHP Artisan** migrate:refresh, s'executarà la creació de totes les migracions amb les creacions de les taules pertinents.

```
+-------------------------+
| Tables_in_mentoringapi  |
+-------------------------+
| materies                |
| migrations              |
| password_resets         |
| users                   |
| usuarios                |
+-------------------------+
```

Imatge model Usuari

```php
namespace App;

use Illuminate\Database\Eloquent\Model;

class Usuarios extends Model
{
    protected $fillable = ['name','surname','email','description','picture','student','mentor'];
}
```

Imatge controlador Usuari

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Usuarios;
class UsuarioController extends Controller
{
    public function index(){
       return Usuarios::get();
    }
    public function create(){//}
    public function store(Request $request)
    {
        $usuario = new Usuarios;
        $usuario->name = $request->name;
        $usuario->surname = $request->surname;
        $usuario->pass = $request->pass;
        $usuario->email = $request->email;
        $usuario->save();
        return '[{"status":"true"}]';
    }
    public function show($id){
        return Usuarios::where('id', $id)->get();
    }
    public function edit($id){}
    public function update(Request $request, $id)
    {
      $usuario = Usuarios::find($id);
      $usuario->name = $request->name;
      $usuario->surname = $request->surname;
      $usuario->email = $request->email;
      $usuario->picture = $request->picture;
      $usuario->pass = $request->pass;
      $usuario->description = $request->description;
      $usuario->mentor = $request->mentor;
      $usuario->student = $request->student;
      $usuario->save();
      return '[{"update":"true"}]';
    }
    public function destroy($id){ }
}
```

Ja hem el controladors, que ens faran la lògica de programació, que inserir hi ha on i el model que s'encarrega de l'inserció a la base de dades, només ens faltarà configurar l'arxiu routes, de api per tal de poder llegir les dades que l'App ens enviarà.

Imatge de les routes donades d'alta i obtingudes mitjançant la comanda php artisan route:list

URI	Name	Action
/		Closure
api/loginMentoring	loginMentoring.store	App\Http\Controllers\LoginMentoringController@store
api/loginMentoring	loginMentoring.index	App\Http\Controllers\LoginMentoringController@index
api/loginMentoring/create	loginMentoring.create	App\Http\Controllers\LoginMentoringController@create
api/loginMentoring/{loginMentoring}	loginMentoring.show	App\Http\Controllers\LoginMentoringController@show
api/loginMentoring/{loginMentoring}	loginMentoring.update	App\Http\Controllers\LoginMentoringController@update
api/loginMentoring/{loginMentoring}	loginMentoring.destroy	App\Http\Controllers\LoginMentoringController@destroy
api/loginMentoring/{loginMentoring}/edit	loginMentoring.edit	App\Http\Controllers\LoginMentoringController@edit
api/materies	materies.index	App\Http\Controllers\MateriaController@index
api/materies	materies.store	App\Http\Controllers\MateriaController@store
api/materies/create	materies.create	App\Http\Controllers\MateriaController@create
api/materies/{matery}	materies.show	App\Http\Controllers\MateriaController@show
api/materies/{matery}	materies.update	App\Http\Controllers\MateriaController@update
api/materies/{matery}	materies.destroy	App\Http\Controllers\MateriaController@destroy
api/materies/{matery}/edit	materies.edit	App\Http\Controllers\MateriaController@edit
api/user		Closure
api/usuarios	usuarios.index	App\Http\Controllers\UsuarioController@index
api/usuarios	usuarios.store	App\Http\Controllers\UsuarioController@store
api/usuarios/create	usuarios.create	App\Http\Controllers\UsuarioController@create
api/usuarios/{usuario}	usuarios.show	App\Http\Controllers\UsuarioController@show
api/usuarios/{usuario}	usuarios.update	App\Http\Controllers\UsuarioController@update
api/usuarios/{usuario}	usuarios.destroy	App\Http\Controllers\UsuarioController@destroy
api/usuarios/{usuario}/edit	usuarios.edit	App\Http\Controllers\UsuarioController@edit
testGet		App\Http\Controllers\TestController@testeaGet
testPost		App\Http\Controllers\TestController@testeaPost

Finalment i tal com podem veure en la imatge superior crearem tants endpoints com funcionalitats de comunicació entre API i APP necessitem. Aquí podeu veure un llistat dels que estan creats actualment.

Instal·lació de l'entorn de desenvolupament (IDE)

Per tal de poder realitzar l'App en Android cal instal·lar un Entorn de Desenvolupament (IDE) adequat per tal propòsit. En aquest cas he escollit Android Studio v3.14 i v3.2, en els dos ordinadors on s'han desenvolupat o provats l'App.

Per tal de poder instal·lar el paquet he anat a la pàgina de https://developer.android.com/studio/?hl=es-419 des d'on he descarregat l'instalador i le executat des de un entorn sota el sistema operatiu de Windows 10.

Un cop realitzat la instal·lació de l'entorn he copiat el contingut de l'App a la ruta, C:\Users\"UserName"\AndroidStudioProjects des de aquí ja podem obrir automàticament el projecte des de l'Android Studio, ja que és la seva carpeta per defecte a l'hora de deixar els projectes creats amb aquest entorn.

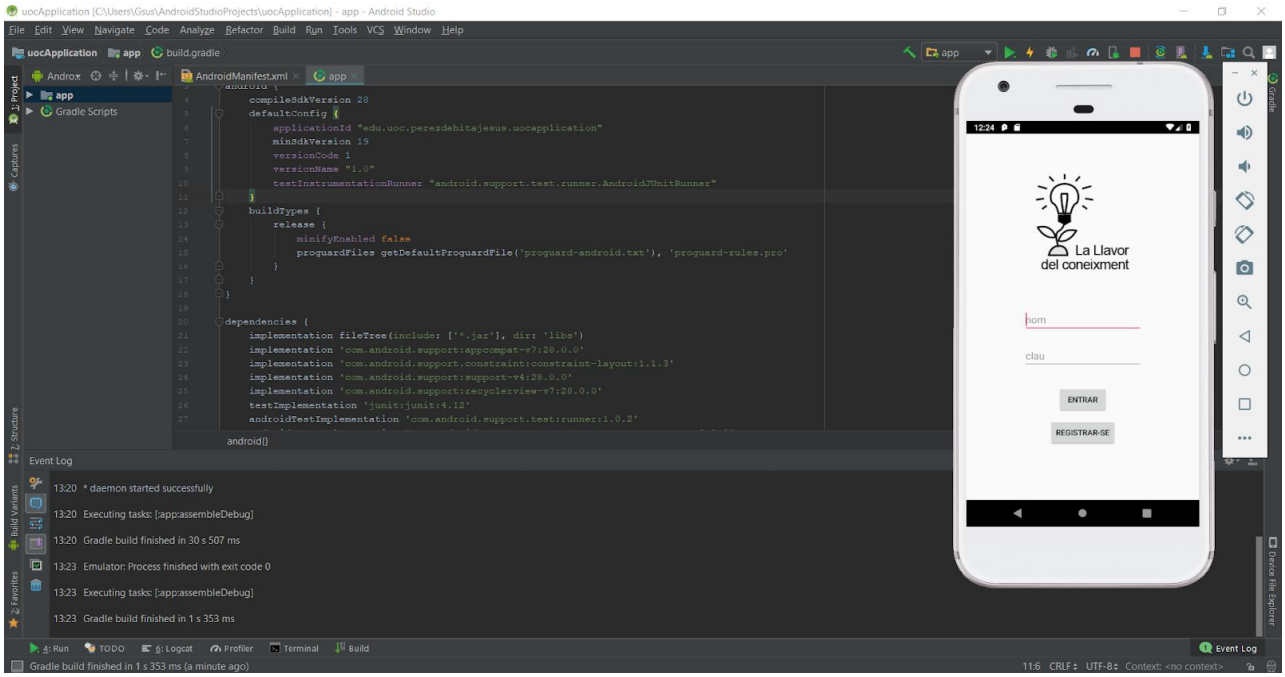
Durant la primer intent de compilació l'Android Studio actualitzarà unes quantes llibreries, normal. En aquest procés trigarà una estona, finalment ens donarà un error en el manifest.xml per tal de treure unes línies que li provoquen conflicte.

Un cop eliminades les línies que ens comenta el propi IDE quedarà de la següent manera.

```
android {
    compileSdkVersion 28
    defaultConfig {
        applicationId "edu.uoc.perezdehitajesus.uocapplication"
        minSdkVersion 19
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

Finalment podem executar amb l'emulador que mantingui una versió mínima de Sdk de 19 com per exemple amb el dispositiu virtual Pixel.

De tal forma que podem veure com queda finalment l'app en el dispositiu virtual.

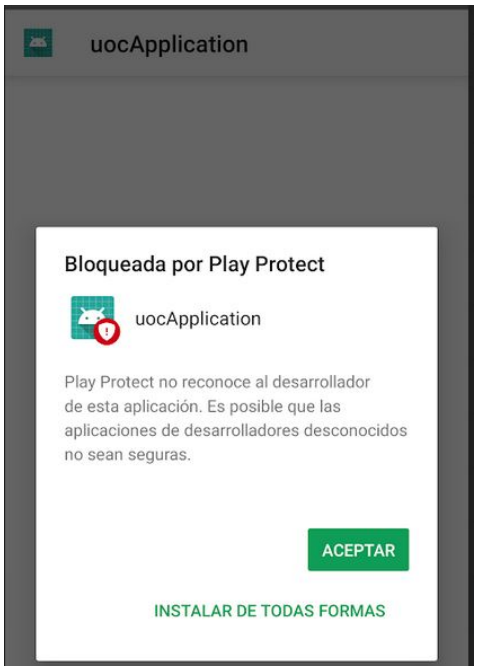


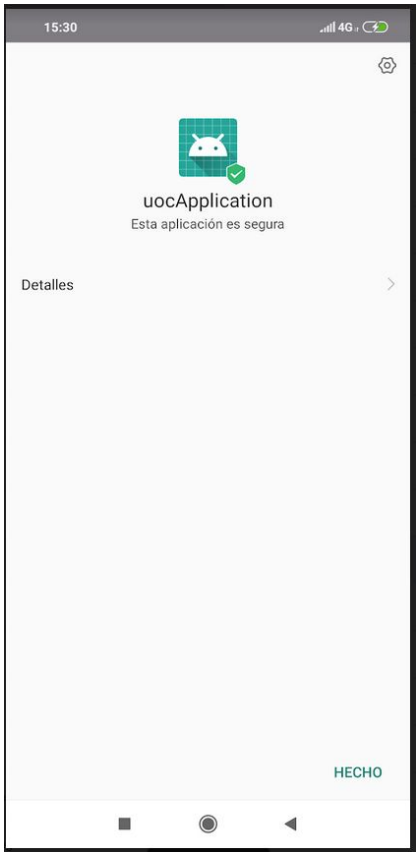
Instal·lació del APK en un dispositiu Mobile

En aquest apartat explicaré detalladament quin son els passos a seguir per tal de poder instal·lar el APK en un mobile.

Primer de tot tenir un dispositiu mòbil amb Android com a Sistema Operatiu

Després permetre-li que instal·li Apps que no siguin d'origen desconeguts, com és el cas.





És probable que ens surti un altre avís d'instal·lació dient que l'app no és del Google Play i que si volem instal·lar-ho igualment.

Un cop acceptat ja tindrem l'App instal·lada en el nostre dispositiu com podem observar a la pàgina superior.

Bibliografía

Laravel Framework for Artisans (2018) web de desenvolupament de Laravel https://laravel.com/
[Accessed 25 nov. 2018]

Android Developer (2018) web de desenvolupament Laravel ttps://developer.android.com/guide/?hl=es-419
[Accessed 27 nov. 2018]

Free Music Archive (2018) http://freemusicarchive.org/music/Scott_Holmes/
[Accessed 15 dec. 2018]

Digital Ocean (2018) https://www.digitalocean.com
[Accessed 20 nov. 2018]

Putty (2018) https://www.putty.org/
[Accessed 10 oct. 2018]