APLICACIÓN MULTIPLATAFORMA PARA FACILITAR LA ADOPCIÓN DE ANIMALES

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Anexo V: Documentación técnica

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Enero de 2019

Autor

Alejandro Barajas González

Tutores

Gabriel Villarrubia González

Juan Francisco De Paz Santana

Contenido

Índice de ilustraciones

1. Introducción

El objetivo de este documento es especificar las técnicas de programación que se han utilizado en este proyecto, para así facilitar próximas modificaciones a otros programadores.

Este proyecto está formado por dos partes bien diferenciadas: la aplicación y el servidor. A continuación, se detallarán como se han desarrollado ambas partes.

En este documento se especifica la programación realizada a través de los entornos de desarrollo utilizados, pero podrían utilizarse otros a los que el programador esté más adaptado.

2. Aplicación móvil

En el desarrollo de la aplicación móvil se ha utilizado Visual Studio y su extensión dedicada a Apache Cordova.

El proyecto se puede abrir utilizando el ejecutable con extensión .sln o importándolo desde la opción Archivo -> Abrir -> Proyecto o solución.

Una vez hecho esto observaremos una estructura similar a la de la imagen:

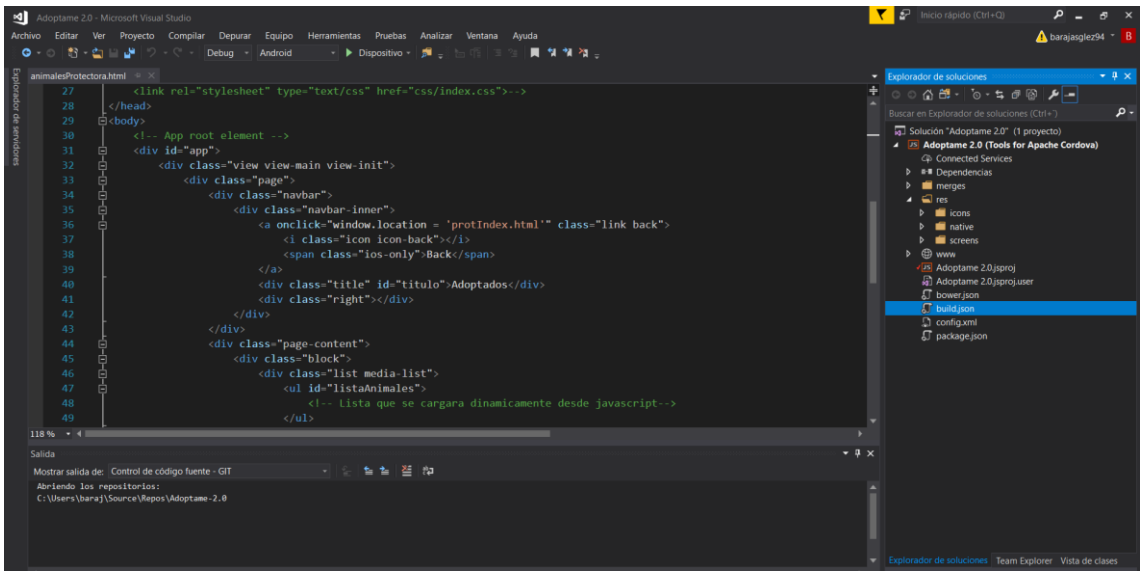

Ilustración 1 - Aspecto del proyecto en Visual Studio

Las partes más importantes a tener en cuenta son: el directorio “res”, el directorio “www” y el archivo “config.xml”.

2.1 Config.xml

En este archivo se puede definir algunas de las propiedades principales de la aplicación, como el nombre, la página de inicio, el idioma, etc. Visual Studio proporciona un asistente gráfico para el manejo de este archivo de manera más sencilla.

También se pueden añadir y quitar complementos que ofrece Apache Cordova para acceder a las funcionalidades del dispositivo en el apartado “Complementos”.

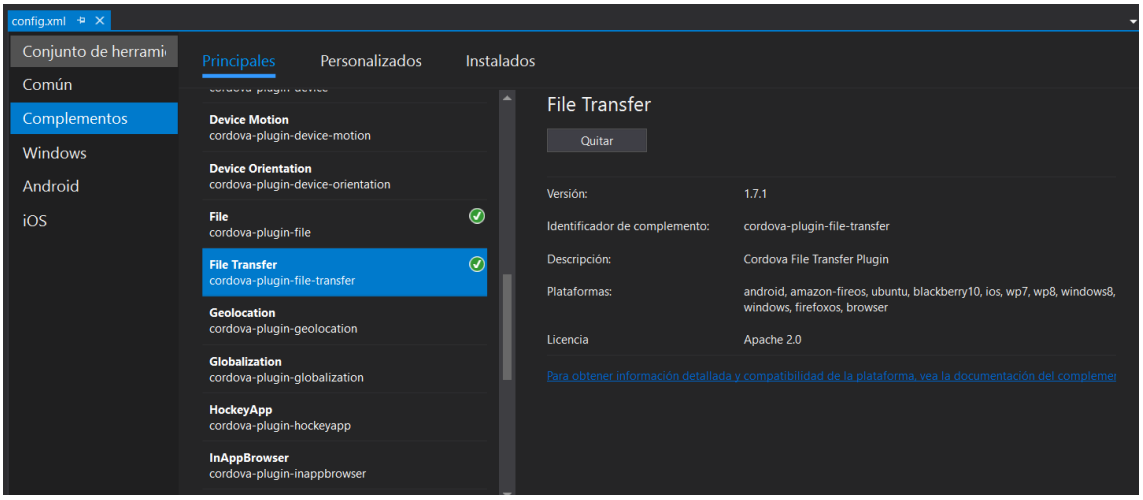

Ilustración 2 - Complementos Apache Cordova

En esta aplicación se utilizan los complementos “File Transfer”, “File” y “Camera”.

Este archivo también se puede modificar a través de código, ya que se trata simplemente de un archivo “.xml”. Este el aspecto que presenta:

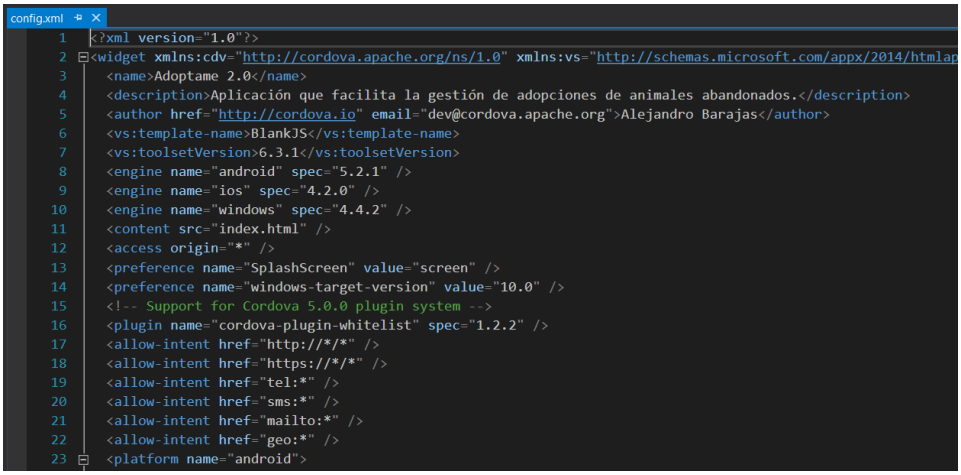

Ilustración 3 - Código config.xml

2.2 Res

Contiene recursos para utilizar en la aplicación, como por ejemplo el propio icono. El propio proyecto genera carpetas diferentes para los distintos sistemas:

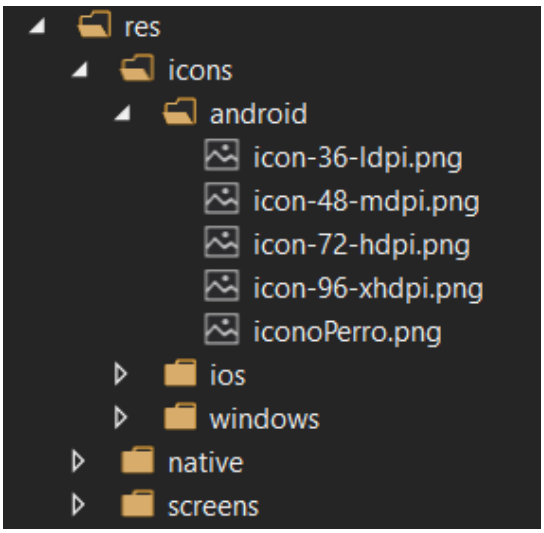

Ilustración 4 - Directorio "res"

2.3 WWW

Es el directorio más importante de todo el proyecto. Contiene todas las pantallas que aparecen en la aplicación, los estilos y scripts. A continuación, se irán detallando todo su contenido.

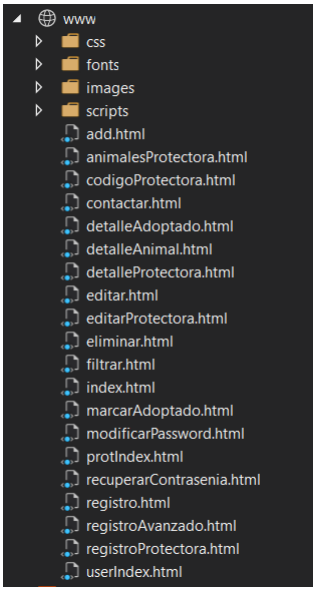

Ilustración 5 – Directorio "WWW"

2.3.1 CSS

En este directorio se almacenan los archivos que dan estilo a las diferentes pantallas de la aplicación:

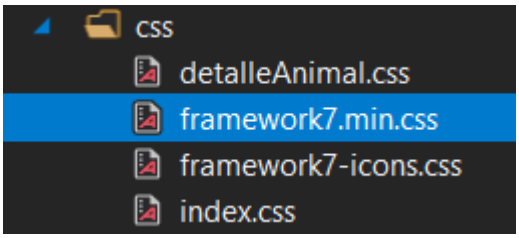

Ilustración 6 - Directorio "CSS"

Los más importantes son los archivos “framework7.min.css” y “framework7-icons.css”

- ***framework7.min.css:*** Contiene los estilos de “Framework 7” en versión minimizada para reducir el tamaño condensando la información, eliminando espacios, retornos de carro, etc…
- ***framework7-icons.css:*** Este archivo se ha importado para dar estilo a los iconos que ofrece Framework 7.

Ambos archivos se han descargado de la página oficial del framework y se han importado manualmente.

2.3.2 Fonts

Este directorio contiene archivos para proveer a “Framework 7” de sus fuentes necesarias.

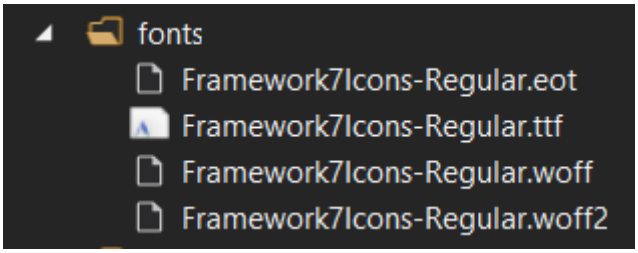

Ilustración 7 - Directorio "Fonts"

2.3.3 Images

Es el directorio en el que se almacenan las imágenes que se introducirán en las diferentes pantallas de la aplicación.

2.3.4 Scripts

Contiene todos los archivos de JavaScript de la aplicación para el manejo de eventos en las diferentes pantalla y los scripts necesarios para utilizar “Framework 7” , “JQuery” y “Chart”.

Se ha generado un archivo por cada pantalla de la aplicación utilizando el mismo nombre. Estos ficheros inicializarán la vista de la aplicación, cargarán datos en la pantalla, manejarán los eventos de los usuarios, etc.

La función más importante que contienen todos los archivos es la siguiente:

```
function onDeviceReady() {
    // Controlar la pausa de Cordova y reanudar eventos
    document.addEventListener('pause', onPause.bind(this), false);
    document.addEventListener('resume', onResume.bind(this), false);

    // TODO: Cordova se ha cargado. Haga aquí las inicializaciones de Cordova y Framework 7.
    app = new Framework7({
        // App root element
        root: '#app',
        // App Name
        name: 'Adoptame2.0',
        // App id
        id: 'adoptame',
        // Enable swipe panel
        panel: {
            swipe: 'left',
        },
        // Add default routes
        routes: [
            {
                path: '/about/',
                url: 'about.html',
            },
        ],
        // ... other parameters
```

Ilustración 8 - Archivo JavaScript

En está función se incluirán la inicialización y carga de controladores de Apache Córdova y la inicialización de la vista de Framework 7 obligatoriamente. También se podrán incluir inicialización de variables, cargar datos que necesite la pantalla, etc.

2.3.5 HTML

Los archivos con extensión ".html" no se encuentran en ningún directorio específico dentro del proyecto.

Estos archivos contienen el código en formato HTML que formarán la parte visual de las diferentes pantallas de la aplicación.

La parte más importante de estos archivos se encuentra en el final, donde se añaden los diferentes scripts que van asociados a cada pantalla.

```
<!-- Path to Framework7 Library JS-->
<script type="text/javascript" src="scripts/framework7.min.js"></script>
<!-- Path to your app js-->
<script type="text/javascript" src="cordova.js"></script>
<script type="text/javascript" src="scripts/platformOverrides.js"></script>
<script type="text/javascript" src="scripts/jquery-3.3.1.min.js"></script>
<script type="text/javascript" src="scripts/index.js"></script>
```

Ilustración 9 - Añadir scripts desde HTML

En la parte superior si se desea se pueden añadir estilos específicos para la pantalla importando archivos CSS.

```
<!-- Path to Framework7 Library CSS -->
<link rel="stylesheet" href="css/framework7.min.css">
<!-- Path to your custom app styles-->
<link rel="stylesheet" type="text/css" href="css/index.css">
```

Ilustración 10 - Aplicar estilos

Todas las pantallas siguen un patrón similar de "divs", ya que lo especifica Framework 7. Para que estas pantallas se muestren es muy importante que se importen bien el script desde donde se inicializa la vista.

3. API REST

Se ha desarrollado una API REST utilizando “SLIM 3”. Para la codificación de las diferentes funciones se ha utilizado “Sublime Text 3”. Este es el aspecto general de la estructura del proyecto:

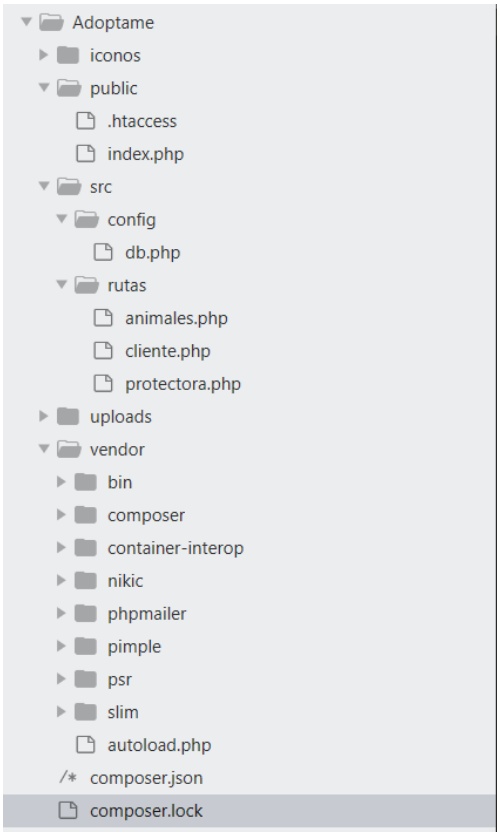

Ilustración 11 - Estructura API REST

A continuación, se detallarán los archivos más importantes.

3.1 Index.php

En este archivo se importarán todos los archivos necesarios para que las diferentes funciones REST sean accesibles a través de los diferentes paths y otras configuraciones necesarias, como por ejemplo el archivo que permite el acceso a la base de datos.

3.2 Db.php

Este archivo contiene la configuración para acceder a la base de datos. Además, se han añadido algunas configuraciones cómo la codificación en UTF 8 y la captura de errores procedente la base de datos.

3.3 Rutas

En este directorio se almacenan los archivos que contienen las funciones que se exponen en la API.

Consta de tres archivos: animales.php, cliente.php y protectora.php que se corresponden con los diferentes paquetes que presenta el sistema (Gestión de animales, Gestión de usuarios y Gestión de protectoras).

```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;

//$app = new \SLIM\App;

use \PHPMailer\PHPMailer\PHPMailer;
use \PHPMailer\PHPMailer\PSMTP;
use \PHPMailer\PHPMailer\Exception;


//---------------------------------------------------------------------
//Obtener todos los usuarios
//---------------------------------------------------------------------
$app->get('/api/clientes',function(Request $request, Response $response){

    //header("Content-Type: application/json; charset=UTF-8");
    $consulta = 'SELECT * FROM usuario';

    try{

        $db = new db();

        //Conexion
        $db = $db->conectar();
        $ejecutar = $db->query($consulta);
        $clientes = $ejecutar->fetchAll(PDO::FETCH_OBJ);
        $db = null;

        //Exportar y mostrar JSON
        echo json_encode($clientes);
```

Ilustración 12 - Ejemplo "cliente.php"

En esta imagen se observa la importación de “PHPMailer” para utilizar las funcionalidades de mail y una de las peticiones para los usuarios.

3.4 Uploads

Contiene las imágenes que se suben desde la aplicación y se utilizaran posteriormente.

3.5 PHPMailer

Es una clase creada específicamente para hacer el envío de emails. Permite enviar emails con archivos adjuntos, introducir código html, estilos, etc.

Esta clase se ha descargado y añadido al proyecto para poder hacer envíos sencillos de emails.

- Fuente: https://github.com/PHPMailer/PHPMailer